VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Monitorovací systém dostupnosti služby pro poskytovatele IP telefonie

Monitoring System of Service Availability for IP Telephony Providers

2014

Bc. Petr Vůjtek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Petr Vůjtek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Monitorovací systém dostupnosti služby pro poskytovatele IP telefonie
Monitoring System of Service Availability for IP Telephony Providers

Zásady pro vypracování:

Cílem diplomové práce je navrhnout a prakticky realizovat monitoring IP telefonů, SIP trunků a dalších komponent. Pro zjištění dostupnosti služby na SIP protokolu bude využíváno metody OPTIONS, rovněž bude automatickým voláním na testovací čísla prověřována funkčnost propojení mezi SIP Proxy. Pro automatické generování volání bude využit open-source generátor sipp. Nedílnou součástí bude notifikace vybraných událostí e-mailem a SMS.

1. Monitoring síťových služeb, protokoly a nástroje Zabbix a Nagios.
2. Dostupnost IP telefonů na protokolu SIP, řešení v Asterisku a pomocí sipsak.
3. Ověřování funkčnosti propojení mezi poskytovateli nástrojem sipp.
4. Návrh a praktická realizace monitorovacího systému s notifikací událostí.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

M. Vozňák,Voice over IP. Vydavatel: VŠB-TU Ostrava, 1. vydání, v Ostravě, 2008, ISBN 978-80-248-1828-3.
L. Madsen, J. Meggelen and R. Bryant, Asterisk: The Definitive Guide. O Reilly, 734 p. , 2011, ISBN 978-14-493-0680-9.
V. Faltinsen, G. Vindheim, Framework conditions and requirements for network monitoring in campus networks. Technical Report GN3-NA3-T4-UFS128, 29 p., TERENA, Amsterdam, 2011.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA OSTRAVA
Fakulta elektrotechniky a informatiky
1

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 2. května 2014 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2. května 2014 .....

Děkuji vedoucímu mé diplomové práce, doc. Ing. Miroslavu Vozňákovi, Ph.D., za konzultace a odborné rady týkající se závěrečné práce. Děkuji katedře telekomunikační techniky za poskytnutí výpočetního výkonu pro virtuální server. A centru informačních technologií VŠB-TUO za poskytnutí služeb univerzitního SMTP serveru a vytvoření alias pro e-mailovou adresu využitou k Zabbix serveru.

Abstrakt

Diplomová práce řeší návrh a realizaci způsobu monitorování služeb a klientských zařízení využívajících SIP protokol. Pro realizaci jsem vybral vhodný dohledový systém a zvolil způsob ověření dostupnosti SIP klientů včetně funkčnosti propojení mezi poskytovateli SIP služeb. Součástí je rovněž monitorování kvality hovoru a praktická realizace monitorovacího systému s notifikací událostí e-mailem a SMS zprávou.

Klíčová slova: diplomová práce, dohledový systém, VoIP, Zabbix, Asterisk, SIPp, SIP

Abstract

The graduation thesis deals with the proposal and realization of the method of monitoring of services and client appliances using SIP protocol. For the realization I have chosen a suitable supervisory system and selected the way of checking accessability of SIP clients including the functionality of connection between SIP service providers. The part is also devoted to monitoring of the quality of conversation and practical realization of the monitoring system with the incident's notification by e-mail and SMS.

Keywords: diploma thesis, monitoring system, VoIP, Zabbix, Asterisk, SIPp, SIP

Seznam použitých zkratek a symbolů

ACELP	–	Algebraic Code-Excited Linear Prediction
AGI	–	Asterisk Gateway Interface
AMI	–	Asterisk Manager Interface
API	–	Application Programming Interface
ARI	–	Asterisk REST Interface
B2BUA	–	Back-to-Back User Agent
CS-ACELP	–	Conjugate Structure ACELP
DAHDI	–	Digium Asterisk Hardware Device Interface
DHCP	–	Dynamic Host Configuration Protokol
DNS	–	Domain Name Server
ENUM	–	E.164 NUmbering Mapping
ETSI	–	European Telecommunications Standards Institute
GSM	–	Global System for Mobile communications
GSM-EFR	–	GSM-Enhanced Full Rate
HTTP	–	Hypertext Transfer Protocol
HP	–	Hewlett-Packard
ICMP	–	Internet Control Message Protocol
IETF	–	Internet Engineering Task Force
IP	–	Internet Protocol
ITU	–	International Telecommunication Union
IVR	–	Interactive Voice Response
ISDN	–	Integrated Services Digital Network
JMX	–	Java Management Extensions
JSON	–	JavaScript Object Notation
MAC	–	Media Access Control
MMUSIC	–	Multiparty Multimedia Session Control
MOS	–	Mean Opinion Score
MP-MLQ	–	Multi-Pulse Maximum Likelihood Quantization
OSI	–	Open Systems Interconnection
PBX	–	Private Branch Exchange
PCM	–	Pulse-Code Modulation
PSTN	–	Public Switched Telephone Network
RAS	–	Registration, Admission and Satus

RPC	–	Remote Procedure Call Protocol
RTP	–	Real-Time Transport Protocol
SDP	–	Session Description Protocol
SIP	–	Session Initiation Protocol
SMS	–	Short Message Service
SMTP	–	Simple Mail Transfer Protocol
SNMP	–	Simple Network Management Protocol
SSL	–	Secure Sockets Layer
TCP	–	Transmission Control Protocol
TDM	–	Time-Division Multiplexing
TLS	–	Transport Layer Security
TTS	–	Text to Speech
UAC	–	User Agent Client
UAS	–	User Agent Server
UDP	–	User Datagram Protocol
URI	–	Uniform Resource Identifier
URL	–	Uniform Resource Locator
USB	–	Universal Serial Bus
VAD	–	Voice Activity Detection
VoIP	–	Voice over Internet Protocol
VŠB-TUO	–	Vysoká škola báňská - Technická univerzita Ostrava
XML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cílem závěrečné práce je návrh a praktická realizace monitorování IP telefonů a SIP komponent. Pro monitorování popisuji v práci dohledové systémy Nagios a Zabbix. Podrobněji jsem popsal systém Zabbix, který jsem ve své závěrečné práci použil k uchovávání informací o monitorovaných prvcích a notifikaci příšlušných uživatelů e-mailem a SMS zprávou. Kromě popisu některých funkcí Zabbix serveru popisuji i Zabbix API a JSON-RPC, které využívám pro komunikaci mé aplikace se Zabbix serverem.

Pro notifikaci SMS zprávami popisuji způsob konfigurace a zapojení GSM modulu.

Další část práce je věnována IP telefonii a protokolům H.323 a SIP. Jelikož v dnešní době převažuje využití SIP protokolu, je jeho popisu věnován větší prostor, zvláště pak SIP metody a odpovědi, které využiji v mé aplikaci. Pro otestování aplikace jsem nainstaloval pobočkovou ústřednu Asterisk a částečně ji nastavil. Asterisk disponuje komunikačním rozhraním AMI, které ve své aplikaci využívám k identifikaci SIP klientů a ověření jejich dostupnosti. Alternativou k ověření dostupnosti jsou testovací nástroje SIP protokolu. Jedním z nich je SIPp, který je vhodný k ověření dostupnosti SIP trunku.

K monitorování IP telefonie nedílně patří i monitorování kvality hovoru. Kvalitu hovoru lze z dostupných informací vypočítat pomocí tzv. E-modelu, který v sobě zahrnuje různé vlivy snižující kvalitu hovoru. Jedním z těchto vlivů je vliv použitého kodeku. Pro zvolené kodeky jsem vypočítal potřebnou šířku pásma a zjistil další potřebné hodnoty vztahující se ke kodeku, které jsou nutné pro výpočet E-modelu.

V další části práce popisuji přípravu testovací topologie, která je částečně na virtuálním serveru katedry telekomunikační techniky. Na virtuálním serveru jsou nainstalovány a spuštěny instance pobočkové ústředny Asterisk a dohledového systému Zabbix. Aplikace, která je výstupem mé závěrečné práce, bude spuštěna na jiném počítači. Na stejném počítači, případně na dalších počítačích, budou spouštěni SIP klienti, které budu v rámci testování monitorovat.

Součástí závěrečné práce je aplikace, jež je rozdělena do tří částí.

První část zajišťuje monitorování SIP klientů, jejichž dostupnost je ověřena pomocí rozhraní AMI pobočkové útředny Asterisk.

V druhé části jsem implementoval mechanismus měření kvality hovoru mezi dvěma segmenty počítačové sítě pomocí datového toku odpovídajícího telefonnímu hovoru, kódovanému zvoleným kodekem. Jako ukazatel kvality simulovaného hovoru jsem použil výpočet hodnoty E-modelu pro čtyři zvolené kodeky.

Poslední část ověřuje dostupnost SIP Proxy pomocí programu SIPp, který odesílá SIP zprávu OPTIONS.

Veškerá data, získaná aplikací, jsou odesílána pomocí programu Zabbix sender do dohledového systému Zabbix, který informace vyhodnocuje a v případě nedostupnosti některého prvku, či snížené kvality simulovaného hovoru, notifikuje uživatele zvoleným způsobem.

2 Monitorovací systémy

Monitorování počítačové sítě je pro provozovatele velmi důležité, protože mu umožňuje rychle reagovat na problémy v síti. Případně jej může monitorovací systém upozornit na aktuální vytížení některého segmentu sítě nebo na nedostupnost některých služeb. Některá řešení monitorovacích systémů umožňují vysokou míru škálovatelnosti, díky rozdělení na moduly, a jsou tedy vhodné pro všechny typy sítí. Mezi takové systémy, z těch bezplatných, patří například Nagios nebo Zabbix. Obě řešení umožňují uživateli implementovat vlastní moduly a rozšířit tak možnosti dohledového systému. Z důvodu vyšší spolehlivosti je vhodné zavést v monitorovacím systému redundanci, díky které by byla včas odhalena nedostupnost primárního monitorovacího systému.

[3, 7]

Po dohodě s vedoucím diplomové práce jsem se zaměřil na monitorovací systém Zabbix, který je podrobněji rozveden v podkapitole 2.2.

2.1 Nagios

Dohledový systém Nagios je v dnešní době velmi oblíbeným open source řešením problematiky monitorování počítačové sítě. Je postaven na principu modulů, které lze volně upravovat, případně implementovat vlastní. Moduly lze vytvářet v různých skriptovacích či programovacích jazycích, jako například C++, shell skript, Perl, Ruby, Python, PHP, C# nebo Java. Nagios je koncipován pro provoz na operačním systému Linux, případně na jiném UNIXovém systému. Pro užívání na operačním systému Windows je nutné použít například mutaci Nagiosu nazvanou Nagwin. [7, 8, 9]

Monitorované objekty se v Nagiosu dělí na dvě kategorie: síťové prvky a služby. Síťové prvky jsou například servery, směrovače, pracovní stanice, tiskárny, VoIP telefony, koncová zařízení atd., zatímco mezi služby řadíme například webové služby, SMTP, FTP, Asterisk apod. Každá služba je sdružena s konkrétním fyzickým zařízením, na kterém je spuštěna. Nagios funguje tak, že v pravidelných intervalech provádí kontrolu definovaných prvků sítě nebo služeb. Nejjednodušším způsobem jak ověřit dostupnost a funkčnost aktivního prvku je pomocí programu ping, využívajícího ICMP zprávy. Pomocí specializovaných modulů může u služeb ověřit jejich reálnou funkčnost. Například webové služby se může dotázat na konkretní URL adresu a ověřit jestli v odpovědi bude kod 200 OK, značící úspěšné vyřízení požadavku.

Silnou stránkou systému Nagios je, že místo konkrétních hodnot, zjištěných monitorováním, používá jen čtyři stavy, interpretující zjištěné hodnoty.

- OK
- WARNING
- CRITICAL
- UNKNOWN

Administrátor se tak neztrácí ve výpisu čísel, ale je informován o abstraktním významu zaznamenaných hodnot. Přehledně vidí vážnost situace a může být informován různým způsobem a s různou intenzitou. Na jednotlivé stavy mohou být vázány odlišné akce. Administrátor může velmi flexibilně konfigurovat způsoby notifikace jednotlivých osob nebo skupin. K notifikaci lze použít e-mail, sms, instant messaging nebo lze pomocí vlastního skriptu notifikovat čímkoliv jiným. Například pomocí technologie TTS je možné převést text do audio souboru, který lze připojit jako média v sestavené audio relaci za pomocí nástroje sipp. Navíc lze notifikaci nastavit tak, aby bylo možné upozornit konkrétního pracovníka zodpovědného za nefunkční prvek, či službu.

Monitorování lze také časově omezit například jen na pracovní dobu nebo jej lze deaktivovat po dobu plánovaného výpadku. Nagios také může uživatele před plánovaným výpadkem upozornit.

S Nagiosem může oprávněná osoba také komunikovat pomocí externích příkazů. Lze tak řešit některé závady vzdáleně, například jejich restartováním. V Nagiosu také můžeme nastavit, aby zaslal příkaz pro restartování prvku automaticky, při zjištění nedostupnosti. Problém tak může být vyřešen bez zásahu administrátora a mnohem rychleji. [9]

Monitorovat VoIP klienty lze v Nagiosu například pomocí modulu *check_calls*, který na klienta zasílá zprávu OPTIONS, jenž je součástí protokolu SIP. Podle odpovědi vyhodnotí dostupnost VoIP klienta. [10]

2.2 Zabbix

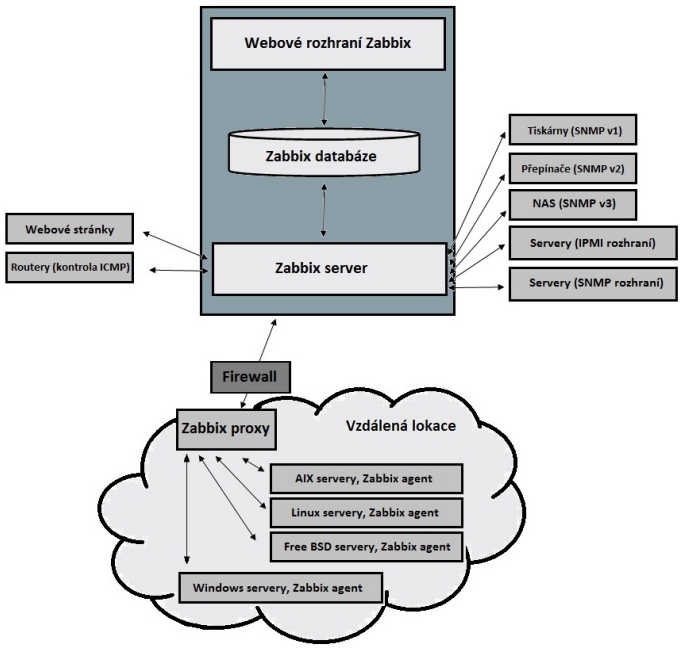
Druhým, v dnešní době používaným bezplatným dohledovým systémem, je Zabbix. Produkt je složen ze dvou částí. První – serverová část probíhá na serveru a je určená pro Unixové operační systémy. Druhou důležitou komponentou je klientská část (agent), která je spuštěna na všech monitorovaných zařízeních. Agent funguje na stejných operačních systémech jako serverová část a navíc podporuje i operační systémy od společnosti Microsoft a Apple. U některých serverů, jako jsou například HTTP či FTP servery, není Zabbix agent nutný. [12]

Systém Zabbix by se dal charakterizovat jako semi-distribuovaný monitorovací systém s centralizovaným řízením. Semi-distribuovaný právě proto, že využívá agenty distribuované na monitorovaných zařízeních. [12]

Mezi výhody dohledového systému Zabbix patří jeho centrální zobrazování informací v uživatelsky snadno použitelném webovém rozhraní, implemetnace serveru jen pro Unixové systémy, které jsou stabilní, což je pro tak klíčový systém velmi důležité. Dále pak nativní Zabbix agenti, vyvíjení zvlášť pro různé verze a unixových systémů. Agent pro operační systémy Windows je společný pro všechny verze. [12]

Zabbix je také schopen monitorovat zařízení používající SNMP (verze 1, 2 a 3) a také zařízení Intelligent Platform Management Interface (IPMI), sloužící jako rozhraní pro monitorování. IPMI je rozhraní, pomocí kterého lze monitorovat hardware zařízení, jeho teplotu a je také schopno zařízení restartovat v případě nefunkčnosti. [11, 12]

V neposlední řadě je užitečnou funkcionalitou vytváření grafů a vizualizací síťové topologie. [12]



Obrázek 1: Příklad nasazení monitorovacího systému Zabbix [12]

2.2.1 Webové rozhraní

Webové rozhraní slouží k přehlednému zobrazení důležitých informací o monitorovaných komponentech. Je napsáno ve skriptovacím jazyce PHP. Pro správné fungování je nutné mít nainstalován PHP alespoň verze 5.1.6 a vhodný webový server, nejlépe Apache. [12, 13]

Pokud vše funguje, tak je webové rozhraní dostupné na adrese: *http://<ip_adresa>/zabbix*. Součástí webového rozhraní, v záložce *Administration/Installation*, je průvodce instalací v němž lze zkontrolovat, zda jsou splněny předepsané prerekvizity. V dalších krocích lze nastavit propojení s databází a upravit nastavení Zabbix serveru. V tomto nastavení je možné zadat i název Zabbix serveru pro případ, že máme spuštěno více instancí. Název se pak objeví v pravém horním rohu každé stránky webového rozhraní a administrátor tak může snadno rozlišit, s kterou instancí právě pracuje. [12]

Základní stránkou webového rozhraní je „Dashboard“, kde jsou zobrazeny ty nejdůležitější informace o sledovaných prvcích. Web je rozdělen do pěti základních kategorií.

První je záložka „Monitoring“, ve které lze vidět data, případné upozornění na problémy a grafy. Další je záložka „Inventory“, ve které lze při správném nastavení vidět soupis dat o monitorovaných prvcích.

V záložce „Reports“ jsou k dispozici oznámení o stavu systému.

Velmi důležitou záložkou je „Configuration“, ve které se provádí veškerá konfigurace monitorovacího systému. Konfiguruje se zde, které prvky mají být monitorovány a jakým způsobem má probíhat notifikace.

Poslední záložkou je „Administration“, ve které lze změnit volby zobrazení, autentizace a je zde i již zmíněná instalace.

[12, 14]

2.2.2 Monitorované prvky

V hierarchii dohledového systému Zabbix je hlavní jednotkou hostitel (angl. Host). Je určen názvem a musí mít definováno některé rozhraní (Agent, SNMP, JMX nebo IPMI). Hostitele pak lze přiřadit do některé skupiny, což následně umožňuje nastavovat například způsob notifikace pro všechny podobné prvky sdružené ve skupině.

Ke každému hostiteli lze přiřadit šablonu obsahující položky, které jsou pro prvky společné. K položkám je pak možné přiřadit trigger, který bude pro všechny prvky vyhodnocovat informace jednotně.

[12, 14]

2.2.3 Šablony

Šablony (angl. Templates) jsou v monitorovacím systému Zabbix důležitou komponentou pro monitorování prvků stejného typu. Umožňují hromadné změny typů informací, které chceme o daném prvku získat.

V šabloně lze, mimo jiné, přidávat položky a typ informací, které bude položka obsahovat.Rovněž lze k šabloně trigger přiřadit, který bude vyhodnocovat údaje přiřazené k položkám.

[12, 14]

2.2.4 Monitorované položky

K monitorovanému prvku lze přiřadit položky, které reprezentují nějakou informaci o prvku. Například ping, verzi systému nebo teplotu procesoru. K prvku lze samozřejmě přiřadit více položek.

Položka je definována svým názvem, klíčem a typem hodnoty. Dále je pak nutné nastavit typ (Zabbix Agent, Zabbix Trapper, SNMP Agent, atd.), kdy pro aktualizaci údajů je pak nutné použít příslušného agenta. V mém případě bude použit program Zabbix sender, takže musí být nastavet typ „Zabbix trapper“. Je nutné také přidat IP adresu, ze které bude Zabbix sender data posílat, mezi povolené adresy v položce „Allowed hosts“.

Typ informace může být číslo, znak nebo text.

[14]

2.2.5 Trigger

Trigger se používá k vyhodnocování údajů o položkách určitého prvku. Jeho výstupem je logická hodnota, buď „True“ nebo „False“. Vyhodnocuje tak, zda vyvolat notifikaci, či ne. Je iniciován změnou údaje nebo po určité době zkontroluje například průměr X posledních hodnot.

Může tak kontrolovat, zda je ping > 100 ms. Pokud je ping vyšší, dojde k vyhodnocení triggeru jako „True“ a je hlášen „PROBLEM“. Na trigger pak může být navázána akce (angl. Action), která odešle e-mail nebo SMS definovaným uživatelům pomocí příslušného média. [14]

2.2.6 Zabbix agent

Zabbix agent je program potřebný k monitorování prvku sofistikovanějším způsobem než jen ICMP. Agent spuštěný na monitorovaném prvku tak může zjišťovat informace o pevném disku, paměti, procesoru, atd. Pomocí agenta může server získávat informace o velikosti volného místa na pevném disku, o vytížení paměti a procesoru a další data, která nejsou běžným monitorováním dostupná.

Při přechodu na vyšší verzi Zabbix serveru není nutné zvyšovat i verze agentů. Server je zpětně kompatibilní se staršími verzemi. Je však doporučeno na novější verze agentů časem přejít, protože mohou poskytovat některé údaje navíc oproti starší verzi. Avšak není zaručena kompatibilita novější verze agenta se starší verzí serveru.

Agenta můžeme, stejně jako server, nainstalovat z repozitáře nebo lze stáhnout zkompilované soubory pro jednotlivé verze operačních systémů. Po instalaci stačí ve webovém rozhraní serveru v záložce „Configuration/hosts“ vytvořit nový monitorovaný prvek, vyplnit název prvku, IP adresu a přiřadit jej k šabloně.

Agent umí pasivní a aktivní kontrolu. Při pasivní kontrole agent vyčkává na vyžádání údajů od Zabbix serveru a až na tuto žádost údaje odešle. Při aktivní kontrole agent nejprve získá ze serveru seznam monitirovaných údajů a poté periodicky odesílá nové údaje na server. Režim agenta lze nastavit v konfiguračním souboru *zabbix_agentd.conf* nastavením parametru *ServerActive*. Hodnotu parametru uvedeme IP adresu Zabbix serveru. Pokud není parametr *ServerActive* uveden, je použita pasivní kontrola.

[12, 14]

2.2.7 Zabbix sender

Alternativou k Zabbix agentu je utilita Zabbix sender, která umožňuje odeslat aktualizované údaje o položkách definovaných v Zabbix serveru. Podmínkou je, aby byla u položky uvedena IP adresa odesílatele v kolonce „Allowed hosts“. Z nepovolených IP adres nejsou data přijímána. Typ položky musí být nastaven jako „Zabbix trapper“.

Program Zabbix sender se spouští z příkazového řádku s příšlušnými parametry. Vybrané parametry, které využiji ve své diplomové práci, uvádím níže.

-z IP adresa nebo hostname Zabbix serveru.

-p Číslo portu, na kterém Zabbix server naslouchá.

-s Název hostitele, vytvořeného v Zabbixu.

-k Název klíče, který je v Zabbixu u hostitele vytvořen.

-o Hodnota klíče.

-i Název textového souboru s názvy hostitelů, klíčů a hodnot.

-I Specifikace zdrojové IP adresy.

Příkladem použití může být následující příkaz.

```
zabbix_sender.exe -z 158.196.244.165 -p 10051 -s 101 -k
   Status -o OK -I 158.196.195.101
```

Zabbix sender je k dispozici pro operační systémy UNIX i Windows.
[14]

2.2.8 Zabbix API

Pro komunikaci mezi Zabbix serverem a aplikacemi třetích stran je vytvořeno API, kterým lze spravovat Zabbix server. Jedná se o textové rozhraní, kdy se přes HTTP posílají zprávy JSON-RPC ve formátu JSON. Prostřednictvím Zabbix API lze přidávat, editovat a mazat veškeré položky, uživatele a oprávnění monitorovacího systému. Ke každému úkonu existuje metoda a její parametry, které jsou ve formátu JSON-RPC odeslány na server, který následně pošle odpověď infikující úspěšné, či neúspěšné provedení metody.

Autentizace probíhá metodou „user.login“, kdy je v parametrech metody odesláno přihlašovací jméno a heslo, které je nastaveno pro přihlášení uživatele přes webové rozhraní. V odpovědi serveru je pak v parametru „auth“ uveden textový řetězec vygenerovaný pro autentizaci uživatele. Tento textový řetězec pak musí být uveden v každé zprávě odeslané přes JSON-RPC.

Pomocí API však nelze aktualizovat hodnoty klíčů hostitelů. K tomu je nutné použít Zabbix Sender popsaný v podkapitole 2.2.7.
[14]

2.2.9 JSON

Zabbix agent komunikuje se serverem pomocí komunikačního protokolu založeného na formátu pro výměnu dat JSON. Jedná se o textový formát pro serializaci strukturovaných dat, který je nezávislý na programovacím jazyce. Umí reprezentovat čtyři základní datové typy (řetězce, čísla, booleovské hodnoty a null) a dvě datové struktury (objekty a pole).

[14, 16]

2.3 SMS brána

K notifikaci příslušné osoby o nedostupnosti některé komponenty počítačové sítě či serveru lze použít e-mail nebo SMS zprávu. V případě nedostupnosti některého klíčového prvku nebo SMTP serveru je nutné použít notifikaci SMS zprávou, u které se předpokládá rychlejší reakce. V případě výpadku SMTP serveru by navíc ani nebylo možné, aby systém někoho upozornil bez SMS brány. [3]

SMS brána je zprostředkovaná například prostřednictvím GSM modemu. Modem lze k serveru, na kterém je spuštěn dohledový systém, připojit pomocí RS232 nebo USB. K serveru lze také připojit mobilní telefon. Komunikace s SMS bránou probíhá pomocí AT příkazů.

Dle dokumentace dohledového systému Zabbix byly testovány tyto GSM modemy:

- Siemens MC35
- Teltonika ModemCOM/G10

Při realizaci jsem otestoval funkčnost zapojení GSM modemu do USB rozhraní:

- Huawei E303

V Linuxu je nutné, aby měl Zabbix oprávnění k zápisu na rozhraní, ke kterému je GSM modem připojen.

[12, 14]

3 IP telefonie

IP telefonie využívá rozvoje rychlosti a datové propustnosti současných počítačových sítí k přenosu hlasu v reálném čase. Prvním standardem pro tento účel se stal H.323 navržený organizací ITU-T v roce 1996. O tři roky později vznikl standard SIP od IETF, který je dnes ve VoIP komunikaci jednoznačně dominantím.

[1]

3.1 H.323

K přenosu hovoru v H.323 jsou využity RTP a RTCP protokoly. RTP přenáší samotný hovor a RTCP má na starosti přenos řídících a stavových informací. Signalizace je přenášena protokoly Q.931 a H.245 přes TCP a protokolem RAS.

Prvky H.323 sítě jsou koncové body a řídící prvky (Gatekeeper). Koncové body se registrují na GK, který tak tvoří zónu spravovanou tímto GK.

[1]

3.2 SIP

SIP je signalizační protokol na aplikační vrstvě sloužící k sestavení, modifikování a ukončení relace s jedním, či více účastníky. Nejčastěji je používám pro audio. Je vyvíjen od roku 1996 pracovní skupinou MMUSIC, která je součástí IETF. MMUSIC v roce 1999 navrhla standard, který byl v březnu 1999 přijat jako RFC 2543. Ještě v roce 1999 vznikla v rámci IETF pracovní skupina nazvaná SIP, která převzala vývoj hlavního jádra protokolu SIP. Na základě jejich práce byl v roce 2002 přijat dokument RFC 3261 popisující SIP. SIP byl navržen tak, aby byl flexibilní, bylo jej možné snadno rozšířit, a aby se snadno implementoval.

Veškerá logika protokolu je vložena do koncových zařízení, která znají i jednotlivé stavy komunikace. Průběh komunikace lze popsat stavovým diagramem. Komunikaci tvoří dialogy (jednotlivá spojení) a dialog je tvořen transakcemi (žádosti a odpovědi). Decentralizovanost protokolu zvyšuje jeho odolnost vůči chybám, ovšem vyžaduje vyšší režii hlaviček zpráv. SIP je tedy navržen zcela odlišně od PSTN sítě, kde je logika uložena v síti a koncová zařízení mohou být primitivní. Cílem při vývoji protokolu SIP bylo vyrovnat se ve funkcionalitě možnostem sítě PSTN, avšak SIP možnosti sítě PSTn překonává díky end-to-end návrhu. Poskytuje tak snadnou implementaci nových služeb, které by v klasické PSTN byly jen obtížně implementovatelné. [1, 17, 19]

Pro audiovizuální komunikaci jen samotný SIP nestačí, a proto je nutné společně s ním použít RTP k přenosu obsahu a SDP pro popis přenášeného obsahu.

Protokol SIP je textově orientovaný inspirovaný HTTP a SMTP protokoly. Osvědčený model komunikace činí ze SIPu robustní a nadčasový protokol. Inspirace pocházející z HTTP jsou například zasílání požadavků z klienta na server, který vrací odpovědi a schéma jejich číselné reprezentace. Protokolem SMTP jsou inspirovány položky *From*, *To* nebo *Subject* v hlavičkách.

SIP entity jsou součástí domény spravované SIP Proxy. Mezidoménová komunikace probíhá mezi různými SIP Proxy. Výjímku tvoří tzv. multidoménové SIP Proxy. SIP entita je jednoznačně identifikovaná jmenným identifikátorem SIP URI. Její obecný tvar je: [1, 19]

```
sip:user:password@host:port;uri-parameters?headers
```

SIP URI je složeno z pole *user*, které identifikuje uživatele a z pole *host* identifikující doménu či hostitele, poskytujícího uživateli prostředky k zajištění komunikace. Pole *password* se dle RFC 3261 nedoporučuje používat, protože autentizace neprobíhá šifrovaně a heslo by tak mohlo být snadno odchyceno. Standardní port pro SIP je 5060 na UDP. Další parametry jsou oddělovány středníkem a jsou-li nutné parametry hlavičky, pak se uvádějí za otazníkem. Ovšem typická SIP URI adresa je v podobě jednoduché konstrukce *sip:user@host*. Příkladem takové adresy může být: [1, 19]

- sip:alice@atlanta.com
- sip:558955099@cesnet.cz

Pomocí záznamů NAPTR v DNS lze zajistit mapování telefonních čísel (ITU-T E.164) na URI, což je technika známá pod označením ENUM. Tímto způsobem lze prostor telefonních čísel začlenit do DNS a provázat jej s jmennými identifikátory, což platí obecně a ENUM tak není jen o SIP. Kupříkladu je možné v DNS namapovat telefonní číslo na URL webové služby a zadáním telefonního čísla univerzity +420596991111 do prohlížeče se tak lze dostat na stránky www.vsb.cz. [1]

3.2.1 Prvky

Základním prvkem SIP komunikace je koncové zařízení označované jako SIP user agent (UA). V nejjednodušší konfiguraci bude komunikace fungovat i mezi dvěma koncovými terminály zasílajícími si navzdájem SIP zprávy.

Při klasickém nasazení SIP řešení bude nutné použít druhý základní prvek označovaný jako SIP server. Mohou jej tvořit SIP Proxy, registrar, redirect a location servery. Tyto servery mohou být provozovány buď na společném hardware nebo zcela odděleně, protože SIP architektura umožňuje jejich úplnou dekompozici. Druhá varianta provozu je vhodná velká řešení, kde distribuovaná architektura umožňuje vyšší robustnost SIP serveru.

Koncové terminály, ve kterých vzniká a je ukončována SIP relace jsou obvykle SIP telefony nebo softwareové aplikace. Nazývají se „user agent“ a mohou jimi být IP telefony, aplikace na počítači, aplikace v chytrém telefonu, PSTN brány, IVR systémy a další. Každý „user agent“ tvoří „user agent server“ a „user agent client“. Zařízení inicializující SIP transakci je v tu chvíli klient a druhá strana server. Během dialogu se tato role mění podle toho, která strana inicializuje transakci.

- UAC odesílá požadavky a přijímá odpovědi
- UAS přijímá požadavky a odesílá odpovědi

SIP Proxy server může být stavový (stateful) nebo bezestavový (stateless). Bezestavové SIP Proxy jen primitivně přeposílají SIP zprávy bez závislostí na jejich vzájemné vazby. Jejich jedinou výhodou je vyšší rychlost oproti stavovým SIP Proxy serverům. Využívají se například jako balanční servery.

Stavové SIP Proxy servery zprávu příjmou, vytvoří si záznam stavu a udržují důležité informace do ukončení transakce nebo dialogu. Inicializují vznik nové zprávy, kterou posílají směrem k příjemci. Mohou tak například větvit nebo přesměrovat spojení. Existují dva typy SIP Proxy serverů. Transakční udržují stav žádosti, dokud není transakce definitivně dokončena a dialogové, které udržují stav dialogu dokud neskončí celé spojení.

Stavové proxy servery mají nižší výkon způsobený tím, že některé transakce mohou trvat dlouhou dobu, po kterou musí SIP Proxy server udržovat stav. Typickou zprávou, která vytváří dlouhodobou transakci, je zpráva INVITE. Většina SIP Proxy serverů je stavových.

Kromě SIP Proxy serveru existují ještě **Redirect server**, který zajišťuje přesměrování spojení a vrací nové URI uživatele. **Registrar server**, jenž přijímá požadavky k registraci, zajišťuje aktualizaci lokalizační databáze a mapuje URI uživatele na URI zařízení. **Location server**, ve kterém jsou uloženy informace o tom, kde lze uživatele a SIP Proxy servery nalézt. Posledním serverem je **B2BUA**, který je speciální typ UA umístěného v cestě spojení. Vytvářené spojení je na něm ukončeno a B2BUA naváže nové spojení na obě komunikující strany. B2BUA má rozsáhlé možnosti doplňkových služeb pro uživatele. Příkladem B2BUA je Asterisk, který bývá používán jako pobočková ústředna.

[1, 19]

3.2.2 SIP Metody

Komunikace probíhá pomocí zpráv, přenášených samostatnými UDP datagramy. Zpráva je tvořena hlavičkou zprávy a může obsahovat i tělo zprávy. V těle zprávy je, většinou pomocí SDP, popis medií. Hlavička je od těla oddělena prázdným řádkem CLRF.

V RFC 3261 jsou definovány základní metody SIP protokolu.

REGISTER je žádost o registraci nebo odregistrování SIP uživatele z registrar serveru.

INVITE metoda inicializující spojení mězi dvěma koncovými uživateli, či změnu parametrů již probíhajícího spojení metodou re-INVITE.

ACK potvrzuje přijetí konečné odpovědi na žádost metodou INVITE.

CANCEL metoda používaná ke zrušení sestavovaného spojení dokud volaný nepotvrdil konečnou odpovědí na žádost INVITE.

OPTIONS touto speciální metodou lze zjistit vlastnosti SIP zařízení. Jedná se o obdobu metody INVITE, kdy však po odpovědi nedojde sestavení spojení. Pomocí této metody lze periodicky zjišťovat podporované funkce a dostupnost SIP koncových zařízení.

BYE metoda používaná k ukončení již sestaveného spojení.

Dodatečně byly v dalších RFC dokumentech specifikujících SIP protokol definovány další metody.

INFO metoda slouží k výměně informací v průběhu spojení. Nezasahuje do stavu hovoru.

PRACK umožňuje spolehlivé doručení a potvrzení dočasné odpovědi.

SUBSCRIBE používá se k přihlášení a odhlášení odběru notifikací o událostech. Notifikace probíhají metodou NOTIFY.

NOTIFY je zpráva odeslaná na základě příhlášení UA k odběru notifikací.

UPDATE může měnit parametry ještě před dokončením metody INVITE.

MESSAGE posílá textové zprávy mezi uživateli o maximální velikosti 1300 bajtů.

REFER používá se k předání hovoru třetí stranou. Například vytáčení kliknutím na webu.

PUBLISH umožňuje skrze dodatečný framework informovat okolí o stavu přítomnosti uživatele.

[1, 17, 18, 19, 20, 21, 22, 23, 24, 26]

3.2.3 SIP odpovědi

Odeslání metody představuje v komunikaci žádost a na ni musí následovat odpověď. Kromě metody ACK, na kterou odpověď již nenásleduje. Pro odpovědi v SIP protokolu byl využit již osvědčený systém kódů z protokolu HTTP. Kód je celé číslo od 100 do 699.

Podle číslice na první pozici se odpovědi rozdělují na informativní (1xx) a konečné (2xx - 6xx).

- 1xx jsou kódy pro dočasné informativní odpovědi. Například kód 180 (Ringing) informuje protistranu o započetí vyzvánění.
- 2xx jsou pozitivní konečné odpovědi. Nejčastěji se jedná o kód 200 OK, který potvrzuje přijetí žádosti metody INVITE.
- 3xx jsou používány při přesměrování hovoru.
- 4xx jsou negativní konečné odpovědi indikující problém na klientské straně.
- 5xx jsou negativní konečné odpovědi indikující problém na straně serveru.
- 6xx jsou kódy indikující globální chybu.

[1, 19]

3.3 Asterisk

Asterisk je otevřený software sloužící jako telefonní ústředna pro IP telefonii. Vznikl v roce 1999 a v současnosti je vyvíjen firmou Digium. Nejnovější verze je Asterisk 12. Je koncipován tak, aby jej bylo možné nainstalovat na klasický stolní počítač s operačním systémem Linux a po připojení odpovídajícího rozhraní může být použit jako pobočková telefonní ústředna (PBX). Jeho architektura je založena na modulech zajišťujících vysokou flexibilitu a škálovatelnost Asterisku. Se správnými moduly a rozhraními lze Asterisk propojit s různými telefonními sítěmi například PSTN a s protokoly IP telefonie SIP i H.323. Asterisk používá od verze 1.4 k vytvoření rozhraní s TDM/PSTN sítí DAHDI.

Konfigurace Asterisk probíhá editacemi konfiguračních souborů s příponou „.conf“, které jsou umístěny ve složce */etc/asterisk*.

[1, 2]

3.3.1 Komunikační rozhraní

Asterisk disponuje několika komunikačními rozhraními pro oboustrannou komunikaci s aplikacemi třetích stran. Jsou to AGI, AMI a ARI. Pro svou diplomovou práci budu využívat rozhraní AMI. Jedná se o textový protokol, kterým lze po autentizaci získat aktuální informace o ústředně. Komunikace může probíhat pomocí HTTP POST nebo zabezpečeně pomocí TLS a OpenSSL, či pomocí TCP a programu Telnet. Přístup přes TCP je nativní a asynchronní. U HTTP je však tato asynchronnost narušena tím, že odpověď je odesílána na základě požadavku. Nativním připojením přes TCP tak může Asterisk přes AMI informovat o událostech i bez přímého požadavku.

Nastavení přístupu přes AMI je nutné explicitně nastavit v souboru */etc/asterisk/manager.conf*. Pro přístup přes HTTP je navíc nutné editovat i soubor */etc/asterisk/http.conf*.

[2, 6]

3.4 Testovací nástroje

Existuje celá řada testovacích nástrojů SIP protokolu. Na základě doporučení vedoucího mé závěrečné práce jsem se zaměřil na nástroje SIPSak a SIPp.

3.4.1 SIPSak

SIPSak je testovací utilita generující SIP provoz. Umožňuje tak simulovat provozní zátěž SIP serverů a koncových zařízení generováním velkého množství SIP požadavků. SIPSak ve verzi 0.9.6 je k dispozici v repozitářích Debian.

[27]

3.4.2 SIPp

SIPp je rovněž testovací zátěžová utilita SIP protokolu vyvinutá techniky společnosti Hewlett-Packard, avšak v současné době se již HP na projektu nepodílí. SIPp dokáže navazovat a ukončovat více souběžných hovorů v síti generováním metod INVITE a BYE. Lze také vytvářet XML scénáře, podle kterých budou testovací hovory generovány. V průběhu testování jsou dynamicky zobrazovány statistiky probíhajících testování. Statistiky obsahují údaje:

- Call rate - počet hovorů za vteřinu.
- Round trip delay - zpoždění mezi odesláním zprávy a přijetím odpovědi.
- Statistika přijatých zpráv.
- Celkový počet hovorů.
- Název serveru.

SIPp může být použito k testování SIP prvků jako například SIP Proxy, B2BUA, SIP media server, SIPx brány a SIP PBX.

Je k dispozici ve verzi pro operační systémy Linux i Windows. Pro Linux jsou ke stažení nejnovější zdrojové soubory verze 3.3 v C++ , které je nutné zkompilovat. V repozitáři Debianu je verze 3.2 pod názvem „sip-tester“. Pro Windows je SIPp prostřednictvím kolekce Cygwin ve verzi 3.2.

Popis parametrů, které použiji:

-sn Role instance SIPp - „uac“ pro klienta a „uas“ pro server. Následuje IP adresa serveru.

-sf Soubor s XML scénářem.

-i Lokální IP adresa.

-m Ukončí testování po zvoleném počtu hovorů.

[28]

4 Kvalita hovoru

Pro měření kvality telefonního hovoru existuje několik metod. První metodou je takzvaný E-model definovaný v doporučení ITU-T G.107.

4.1 E-model

E-model, kde E znamená „Ear“, byl definován v doporučení ITU-T G.107 v roce 1998 . Nejnovější verze doporučení je z roku 2011. Kombinuje vlivy několika různých parametrů přenosu ovlivňujících výslednou kvalitu hovoru. Pro výpočet míry kvality hovoru je odvozen vzorec 1.

$$R = R_o - I_s - I_d - I_{e-eff} + A \tag{1}$$

R ve vzorci vyjadřuje rating nebo-li ohodnocení kvality hovoru. Nabývá většinou hodnot od 50 do 100 viz tabulka 2. Při hodnotách R nižších než 50 je kvalita hlasového přenosu zcela nevhodná.

R_0 vyjadřuje základní poměr signálu a šumu včetně dalších zdrojů šumu jakými jsou obvodový šum a šum okolí. Pro zjednodušený E-model je konstantní R_0 = 94,7688.

I_s je faktor kombinující nedílný šum, který se více či méně simultánně vyskytuje společně s užitečným signálem. Pro zjednodušený E-model je konstantní I_s = 1,4136.

I_d reprezentuje zhoršení způsobené zpožděním včetně zpoždění ozvěn. Pro zpoždění nižší než 100ms je $I_d = 0$.

I_{e-eff} udává zhoršení způsobené použitým kodekem. Vypočítá se rovnicí 2.

$$I_{e-eff} = I_e + (95 - I_e) \cdot \frac{P_{pl}}{\frac{P_{pl}}{BurstR} + B_{pl}} \tag{2}$$

P_{pl} je procentuální ztrátovost paketů.

B_{pl} udává robustnost použitého kodeku.

BurstR vyjadřuje, zda je ztrátovost paketů náhodně rozložena v čase, nebo zda jsou ztráty shlukovány.

A je faktor zvýhodnění na základě soustředění posluchače. Příklady zvýhodnění jsou uvedeny v tabulce 1.

[30, 31, 32, 33, 34]

Situace, ve které je posluchač při komunikaci	Maximální hodnota A
Běžné kabelové spojení	0
Pohyb po budově a spojení mobilní sítí	5
Pohyb v terénu nebo ve vozidle	10
Spojení do těžko přístupných míst například pomocí satelitního spojení	20

Tabulka 1: Příklady zvýhodnění faktorem A [30]

MOS	Kvalita hovoru	R-faktor	Parametr MOS	Subjektivní spokojenost
5	Výborná	100 - 90	4,50 - 4,34	Velmi spokojený
4	Dobrá	90 - 80	4,34 - 4,03	Spokojený
3	Průměrná	80 - 70	4,03 - 3,60	Někteří uživatelé nespokojeni
2	Špatná	70 - 60	3,60 - 3,10	Mnoho uživatelů nespokejno
1	Nedostatečná	60 - 50	3,10 - 2,58	Téměř všichni nespokojeni

Tabulka 2: Hodnocení kvality hovoru MOS, R-faktor [30]

4.2 MOS

Pro subjektivní hodnocení kvality hovoru posluchačem byla v doporučení ITU-T P.800 zavedena stupnice MOS, která je uvedena v tabulce 2. Kvalita hovoru je v tomto případě hodnocena posluchačem účastnícím se experimentu. Posluchač je uzavřen v místnosti definovaných rozměrů a parametrů a je mu pouštěn záznam hovoru. Posluchač poté podle svého subjektivního názoru určí na stupnici 1 - 5 kvalitu hovoru. Experimentu se účastní více osob, kterým je za stejných podmínek pouštěna stejná nahrávka. Aritmetickým průměrem všech hodnocení je pak určena kvalita přenosu hovoru dle stupnice MOS. [31, 32, 35, 38]

Faktor R, vypočtený pomocí E-modelu, lze převést na hodnotu MOS stupnice převodem uvedeným rovnicí 3.

$$\begin{array}{ll} pro\ R < 0: & MOS = 1 \\ pro\ 0 < R < 100: & MOS = 1 + 0,035R + R(R-60)(100-R)7 \cdot 10^{-6} \\ pro\ R > 100: & MOS = 4,5 \end{array} \tag{3}$$

[30, 31, 32, 34, 35]

4.3 Kodeky

Kvalita hovoru je také ovlivněna zvoleným kodekem použitým ke kompresi a dekompresi hovoru. V E-modelu je zohledněna parametrem I_{e-eff}. Pro každý kodek je v doporučení ITU-T G.113 uvedena hodnota parametru I_e, kdy s rostoucí hodnotou klesá kvalita hovoru zvuku hovoru. Avšak je to kompenzování nižší přenosovou rychlostí potřebnou k přenosu komprimovaného hovoru. Dalším uvedeným parametrem pro ka-

Typ kodeku	Reference	Bitový tok [kbit/s]	I_e	B_{pl}
PCM	ITU-T G.711	64	0	4,3
MP-MLQ	ITU-T G.723.1	6,3	15	16,1
ACELP	GSM 06.60	12,2	5	10,0
CS-ACELP	ITU-T G.729-A + VAD	8	11	19,0

Tabulka 3: Doporučené hodnoty E-modelu pro kodeky [1, 32, 37]

ždý kodek je B_{pl} udávající odolnost kodeku proti ztrátám při přenosu. Vybrané kodeky a potřebné parametry jsou uvedeny v tabulce 3.

[30, 37]

- PCM je v současné době nejpoužívanější kodek definovaný v doporučení ITU-T G.711. Kvalita přenášeného hlasu je totožná s kvalitou hlasu při běžném telefonním hovoru. Bitový tok na výstupu kodéru je 64 kbit/s. Signál je při kódování vzorkován na 8 kHz a osmibitové sekvence. Rámec trvá 20 ms.
- MP-MLQ používá například doporučení ITU-T G.723.1. Bitový tok na výstupu kodéru je 6,3 kbit/s. Jeden rámec trvá 30 ms.
- Kódování ACELP používá GSM-EFR kodek specifikovaný ETSI GSM 06.60. Bitový tok je 12,2 kbit/s. Jeden rámec má 244 bitů. Při uvedeném bitovém toku pak jeden rámec trvá 20 ms. [32, 39]
- CS-ACELP - doporučení ITU-T G.729. Bitový tok je 8 kbit/s a rámec trvá 10 ms.

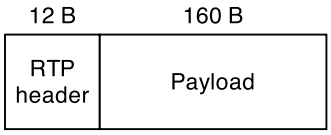
Pro simulaci hovoru kódovaného některým z uvedených kodeků je nutné generovat datový tok odpovídající danému kodeku. U každého kodeku je uveden bitový tok určující jakou šířku pásma kodek zabírá. Jedná se však o údaj vztahující se čistě k výstupu s kodéru. Pro přenos počítačovou sítí je nutné data rozdělit do RTP paketů a ty postupně enkapsulovat. V každém kroku dojde k přidání hlavičky a u ethernetového rámce navíc i k přidání patičky. Tím narůstá bitový tok a tím i potřebná šířka pásma pro přenos dat. Šířku pásma lze obecně vypočítat vztahem 4.[1]

$$BW_M = \sum_{i=1}^{M} \frac{S_{Fi}}{\Delta t_i} \tag{4}$$

V aplikaci, která bude počítat kvalitu simulovaného hovoru, bude dostačující připočítat k užitečné zátěži RTP hlavičku. Hlavičky ostatních vrstev ISO/OSI modelu jsou přidány standardním průchodem počítačovou sítí.

Bitovou velikost RTP paketu (S_{AL}) lze vypočítat sečtením užitečné zátěže (P_S) a 12 bajtové RTP hlavičky (H_{RTP}).

$$S_{AL} = H_{RTP} + P_S \tag{5}$$



Obrázek 2: RTP paket

Typ kodeku	**Reference**	**Velikost [B]**	**Pakety/s**	**Šířka pásma** (L_7) **[kbit/s]**
PCM	ITU-T G.711	172	50	68,8
MP-MLQ	ITU-T G.723.1	36	33	9,5
ACELP	GSM 06.60	31	50	17,2
CS-ACELP	ITU-T G.729-A + VAD	22	100	17,6

Tabulka 4: Přepočtené hodnoty pro RTP pakety [1, 37]

Hodnotu užitečné zátěže (Payload) kodeku lze spočítat ze známých údajů o kodeku. Například pro kodek PCM je znám jeho bitový tok 64 kbit/s a odstup jednotlivých paketů 20 ms. To znamená, že za jednu sekundu se přenese 50 paketů. Z toho lze spočítat, že jeden paket má 1280 bitů, což je 160 bajtů.

$$P_S = \frac{64\,000}{50 \cdot 8} = 160\ [B] \tag{6}$$

Výpočtem rovnice 5 a dosazením výpočtu 6 vychází, že velikost RTP paketu včetně hlavičky je 172 bajtů. Znázornění RTP paketu je na obrázku 2.

Nyní, když je známa velikost dat, které je nutné přenést za jednotku času, lze zpětně dopočítat potřebnou šířku pásma na aplikační vrstvě ISO/OSI modelu (BW_{L7}).

$$BW_{L7} = 172 \cdot 8 \cdot 50 = 68\,800\ [kbit/s] \tag{7}$$

Výpočtem rovnice 7 vyjde šířka pásma pro RTP paket přenášející sekvenci hovoru kódovanou PCM 68,8 kbit/s. Pro simulaci hovoru kódovaného PCM tak bude aplikace generovat bitový tok 68,8 kbit/s. Dodatečné hlavičky vrstev OSI modelu budou připojeny při průchodu sítí.

Přepočítané hodnoty pro vybrané kodeky s RTP hlavičkou jsou uvedeny v tabulce 4.

Na fyzické vrstvě ISO/OSI modelu je pak šířka pásma pro kodek ještě vyšší v důsledku vyššího počtu hlaviček přidaných při enkapsulaci. Toto navýšení může u PCM kodeku činit téměř 50% - 92 kbit/s oproti 64 kbit/s na výstupu z kodéru.

[1]

5 Příprava testovací topologie

Topologie, na které budu praktickou část své závěrečné práce testovat, bude částečně na virtuálním serveru. Na něj jsem nainstaloval operační systém Debian GNU/Linux 6.0.6 Squeeze. V tomto operačním systému jsem nainstaloval dohledový systém Zabbix 2.0.4 a pobočkovou telefonní ústřednu Asterisk 11.

5.1 Monitorovací systém

K monitorování SIP prvků jsem zvolil Zabbix, který bude uchovávat informace o monitorovaných prvcích a rovněž bude zabezpečovat notifikaci uživatelů.

Viz podkapitola 2.2.

5.1.1 Instalace Zabbix

V současné době lze v distribuci operačního systému Linux Debian nainstalovat Zabbix z repozitáře. Jedná se však Zabbix verze 1.8. Tuto verzi lze později přeinstalovat na novější (Zabbix verze 2.0.4).

Příkaz pro instalaci Zabbix z repozitáře:

```
apt-get install zabbix-agent zabbix-frontend-php zabbix-
   server-mysql build-essential libmysqlclient15-dev
   libcurl4-openssl-dev libsnmp-dev snmp snmpd php5-mysql
```

Vytvořím nový adresáře pro Zabbix:

```
mkdir -p /root/src/zabbix
```

Přejdu do adresáře zabbix:

```
cd /root/src/zabbix
```

Pomocí programu wget stáhnu do adresáře zabbix nejnovější verzi Zabbix 2.0.4: [1]

```
wget http://sourceforge.net/projects/zabbix/files/
   ZABBIX%20Latest%20Stable/2.0.4/zabbix-2.0.4.tar.gz/
```

Příkazem tar rozbalím stažený soubor:

```
tar fvxz zabbix-2.0.4.tar.gz
```

Použité atributy:

-f Název archivu.

-v Informování o průběhu dekomprimace.

-x Extrahuje soubory z archivu.

-z Specifikuje kompresní program gzip.

[1] Ke dni 14. dubna 2014 je již k dispozici novější stabilní verze 2.2.3: wget http://sourceforge.net/projects/zabbix/files/ZABBIX Latest Stable/2.2.3/zabbix-2.2.3.tar.gz

Přesunu se do adresáře zabbix-2.0.4:

```
cd zabbix-2.0.4
```

Spustím konfigurační příkaz:

```
./configure --enable-server --enable-agent --with-mysql --
   with-libcurl --with-net-snmp
```

Konfigurační příkaz spouští Zabbix server a Zabbix agenta monitorujícího samotný server. Dále specifikuje použití systému řízení báze dat MySQL. Místo ní lze použít také DB2, Oracle, PostgreSQL nebo SQLite. Dále pak specifikuje použití knihovny libcURL pro práci s cURL a balíček NET-SNMP pro protokol SNMP. Volitelně je možné povolit podporu IPv6 atributem `--enable-ipv6`.

Zkontroluji, zda bylo vše úspěšně konfigurováno. Pokud by se vyskytla chyba, bylo by nutné dohledat chybějící -dev balíčky a poté spustit konfigurační příkaz znovu. Na základě konfiguračního příkazu se vygeneruje validní Makefile a Zabbix je tak připraven ke kompilaci.

Kompilaci provedu příkazem:

```
make install
```

Smažu adresář se starými soubory webového rozhraní Zabbix /usr/share/zabbix, znovu vytvořím prázdný adresář a nakopíruji do něj PHP soubory z novější verze.

```
rm /usr/share/zabbix
mkdir /usr/share/zabbix/
cp -a ./frontends/php/* /usr/share/zabbix/
```

V dalším kroku přejmenuji konfigurační soubor uživatelského rozhraní zabbix.conf.php.example na zabbix.conf.php a upravím v něm parametry pro přístup do MySQL databáze. Při použití databáze MySQL je potřeba v souboru změnit jen heslo pro přístup do databáze.

```
mv /usr/share/zabbix/conf/zabbix.conf.php.example /usr/
   share/zabbix/conf/zabbix.conf.php
nano /usr/share/zabbix/conf/zabbix.conf.php
```

Dále musím změnit oprávnění adresáře obsahujícího konfigurační soubory webového rozhraní, aby do nich měl Apache práva pro zápis.

```
chown -R www-data /usr/share/zabbix/conf
```

Je doporučeno přidat do souboru /etc/services následující porty pro Zabbix.

```
nano /etc/services

zabbix-agent 10050/tcp # Zabbix Agent
zabbix-agent 10050/udp # Zabbix Agent
zabbix-trapper 10051/tcp # Zabbix Trapper
zabbix-trapper 10051/udp # Zabbix Trapper
```

Pomocí phpMyAdmin jsem v MySQL databázi vytvořil uživatele s uživatelským jménem zabbix a heslem, které jsem zadal v konfigurační soubor uživatelského rozhraní zabbix.conf.php, a přidělil jsem mu plná práva do databáze. Vytvořil jsem databázovou strukturu pomocí MySQL klienta.

```
mysql -u'zabbix' -p'zabbixheslo'
create database zabbix character set utf8;
quit;
```

Vytvořím databázi ze souborů verze stažené z repozitáře.

```
cd create/schema
cat mysql.sql | mysql -u'username' -p'password' zabbix
cat images_mysql.sql | mysql -u zabbix -p'password' zabbix
cd ../data
cat data.sql | mysql -u zabbix -p'password' zabbix
```

Ve webovém prohlížeči zobrazím webové rozhraní Zabbix na adrese http://158.196.244.165/zabbix. Na úvodní stránce je vypsán seznam parametrů, které mají být změněny, aby splňovaly minimální požadavky systému Zabbix. V mém případě se jednalo o hodnotu proměnné v souboru /etc/php5/apache2/php.ini post_max_size, která musí být minimálně 16 MB, max_execution_time a max_input_time, které musí být obě minimálně 300 sekund.

Nakonec spustím oba procesy:

```
/etc/init.d/zabbix-server start
/etc/init.d/zabbix-agent start
```

[13]

5.1.2 Konfigurace Zabbix

Konfigurace dohledového systému Zabbix probíhá ve webovém rozhraní, ovšem lze jej konfigurovat i přes Zabbix API pomocí JSON-RPC, které je popsáno v podkapitolách 2.2.8 a 2.2.9.

[14]

5.1.2.1 Struktura monitorovaných prvků Nejvíce monitorovaných prvků budou tvořit SIP klienti, kteří budou do Zabbix serveru přidáváni dynamicky mou aplikací na základě údajů získaných z Asterisk příkazem „SIPpeers“. K nim je však třeba mít připravenou skupinu, do které budou prvky přiřazeny, a šablonu definující monitorované položky (podkapitoly 2.2.2 a 2.2.3).

K jednotlivým položkám jsem vytvořil triggery, které vyhodnocují údaje získané o monitorovaných prvcích.

Nejprve jsem vytvořil skupinu „PhonesGroup“ ve webovém rozhraní v záložce „Configuration\Host groups“. Následně jsem vytvořil šablonu „PhoneTemplate“ v záložce „Configuration\Templates“. V šabloně jsem vytvořil položky typu Zabbix trapper (podkapitola 2.2.4) s následujícími parametry a k nim připojil příslušný trigger:

```
Name: Status
Key: Status
Type of information: Text
Na položku je navázán trigger vyhodnocující, zda je hodnota
    klíče Status = OK

{PhoneTemplate:Status.regexp(OK)}#1

Name: RoundTripDelay
Key: Time
Type of information: Decimal
Units: ms
Na položku je navázán trigger vyhodnocující, zda je doba
   zpoždění < 100 ms

{PhoneTemplate:Time.last(,1)}>100
```

Dalším krokem je vytvoření akce, která na základě vyhodnocení triggeru o nedostupnosti SIP klienta odešle e-mail uživateli. Podmínky pro spuštění akce jsou:

(A) Maintenance status not in maintenance

(B) Trigger value = PROBLEM

(C) Host group = PhonesGroup

Na tyto podmínky je návázána akce odesílající e-mail přes smtp.vsb.cz uživateli Zabbix administrator.

Další částí práce je kontrola kvality hovoru. Aplikace bude simulovat hovor mezi dvěma agenty umístěnými v odlišných segmentech počítačové sítě. Opět dynamicky budou vytvářeny prvky reprezentující linku propojující dva segementy sítě. Vytvořil jsem pro ně skupinu „NetworkSegmentsGroup“ a šablonu „NetworkSegmentsTemplate“, ve které jsou položky pro každý kodek z tabulky 3, se kterým bude proveden simulovaný hovor. Ke každému kodeku pak bude uvedena číselná hodnota R-faktoru výpočtená prostřednictvím E-modelu. Viz podkapitola 4.1.

Na tyto položky bude navázán trigger, vyhodnocující kvalitu hovoru a akce upozorňující na sníženou kvalitu hovoru mezi danými segmenty počítačové sítě. Konkrétně bude trigger ověřovat, zda není hodnota R-faktoru nižší než 80, což je dle tabulky 2 průměrná kvalita hovoru a přepočteno na stupnici MOS a jeho slovní interpretaci jsou „Někteří uživatelé nespokojeni“.

Poslední částí je vytvoření prvku reprezentujícího „SIP Proxy“ a položky budou aplikací dopisovány údaje o dostupnosti SIP Proxy služby. V případě nedostupnosti služby vyvolá trigger akci, odesílající e-mail a SMS uživateli Zabbix administrator.

[12, 14]

5.1.2.2 Notifikace Dohledový systém Zabbix umožňuje notifikaci pomocí SMS zpráv, E-mailů, Jabber IM či vlastního skriptu. Ve webovém rozhraní Zabbix lze v záložce „Administration\Media types“ definovat tyto typy médií.

Pro notifikaci SMS zprávami je nutné připojit k serveru GSM modem. Lze jej připojit k rozhraní RS232 či USB. Ke své práci jsem zvolil GSM modem Huawei E303 do USB rozhraní od O_2 Telefonica. Zařízení jsem připojil ve Windows 7 a namapoval ve VMware k Linuxu nainstalovanému na virtuálním serveru. Zařízení se namapuje jako „ttyUSB0“ a cestu k této složce vyžaduje Zabbix k vytvoření média typu SMS. Parametry nastavení SMS média jsou v mém případě následující:

```
Name: SMS - Huawei
Type: SMS
GSM modem: /dev/ttyUSB0
```

Aby měl Zabbix server oprávnění k přístupu k „ttyUSB0“ je nutné přidat jej například do skupiny, která již k němu má oprávnění, následujícím příkazem:
usermod -a -G groupName userName.

```
usermod –a –G dialout zabbix
```

Nastavení média odesílajícího e-mail se provádí obdobně. Jako typ média se zvolí „Email“ a vyplní se SMTP server, SMTP helo a SMTP e-mail. Protože je můj testovací Zabbix server spuštěn ve školní počítačové síti, nastavil jsem univerzitní SMTP server. SMTP helo se uvádí název domény a SMTP e-mail bude uveden jako e-mailová adresa odesílatele. Po dohodě s Centrem informačních technologií VŠB-TUO jsem získal alias na svoji univerzitní e-mailovou adresu „zabbix-vuj003@vsb.cz“.

```
Name: smtp.vsb.cz
Type: Email
SMTP server: smtp.vsb.cz
SMTP helo: vsb.cz
SMTP email: zabbix-vuj003@vsb.cz
```

Dalším krokem je vytvoření uživatele nebo editace uživatele v záložce „Administration\Users\Media“, kde lze přidat média, pomocí kterých bude notifikace uživatele probíhat. Pro notifikaci SMS zprávami je nutné nastavit telefonní číslo v kolonce „Send to“ nejlépe ve formátu včetně mezinárodní předvolby +420 XXX XXX XXX. Dále pak lze nastavit dny v týdnu a rozmezí hodin, kdy může notifikace probíhat, případně vybrat stavy vážnosti problému vedoucího k notifikaci.

V záložce „Configuration\Actions“ lze vytvořit akci a zvolit podmínky při vyhodnocování získaných dat, kdy bude indikován problém a případně následovat notifikace. V záložce „Operations“ pak jsou definování uživatelé a média, kterými budou o dané situaci informováni.

[12, 14]

5.2 Pobočková ústředna

Jako pobočkovou ústřednu ke zprostředkování IP telefonie pomocí SIP protokolu jsem dle zadání diplomové práce použil Asterisk, který je popsán v podkapitole 3.3. Dále jsem pro testovací účely vybral a nainstaloval dva SIP klienty (Yate a Ekiga).

5.2.1 Instalace Asterisk

Při instalaci z repozitáře Debianu je k dispozici poměrně zastaralá verze Asterisk 1.8.13.1. Nejnovější verzí je Asterisk 12, ovšem nejedná se ještě o zcela stabilní verzi. Z tohoto důvodu jsem se rozhodl nainstalovat Asterisk 11 zkompilováním zdrojových souborů.

Dle návodu je nejprve vhodné operační systém zaktualizovat.

```
apt-get update
apt-get upgrade -y
reboot
```

Dalším krokem je instalace prerekvizit potřebných ke kompilaci.

```
apt-get install build-essential
apt-get install wget
apt-get install libssl-dev
apt-get install libncurses5-dev
apt-get install libnewt-dev
apt-get install libxml2-dev
apt-get install linux-headers-\$(uname -r)
apt-get install libsqlite3-dev
apt-get install uuid-dev
```

Dále je nutné stáhnout zdrojové soubory současných verzí balíčků DAHDI, Asterisk a libpri do složky /usr/src/.

```
cd /usr/src/

wget http://downloads.asterisk.org/pub/telephony/
   dahdi-linux-complete/dahdi-linux-complete-current.tar.gz
wget http://downloads.asterisk.org/pub/telephony/libpri/
   libpri-1.4-current.tar.gz
wget http://downloads.asterisk.org/pub/telephony/asterisk/
   asterisk-11-current.tar.gz
```

Rozbalím všechny tři balíčky.

```
tar zxvf dahdi-linux-complete*
tar zxvf libpri*
tar zxvf asterisk*
```

Instalace DAHDI.

```
cd /usr/src/dahdi-linux-complete*
make
make install
make config
```

Instalace knihovny libpri pro komunikaci přes rozhraní ISDN.

```
cd /usr/src/libpri*
make
make install
```

Jako poslední provedu instalaci samotného Asterisku.

```
cd /usr/src/asterisk*
./configure
make menuselect
make
make install
make config
make samples
```

Po úspěšném nainstalování Asterisku a všech prerekvizit spustím nejprve DAHDI příkazem:

```
/etc/init.d/dahdi start
```

A nakonec spustím Asterisk příkazem:

```
/etc/init.d/asterisk start
asterisk -rvvv
```

[29]

5.2.2 Konfigurace Asterisk

Základní nastavení pobočkové ústředny Asterisk se provádí editací konfiguračních souborů. Pro účely testovací topologie jsem upravil soubor */etc/asterisk/sip.conf*. Následujících několik řádků definuje dva SIP klienty, kteří se mohou v ústředně registrovat.

```
[101] ;Název
disallow=all ;Zakazuje všechny kodeky
host=dynamic ;Klient se může připojit z libovolné IP adresy
type=friend ;Definování typu připojení
dtmfmode=rfc2833
allow=alaw ;Specifikovaný kodek G.711 A-law
qualify=yes
canreinvite=yes
insecure=port
context=dial-local

[102]
disallow=all
host=dynamic
type=friend
dtmfmode=rfc2833
allow=alaw
qualify=yes
canreinvite=yes
insecure=port
context=dial-local
```

V souboru */etc/asterisk/extensions.conf*, který definuje dial plan, tedy soubor pravidel určujících například chování ústředny při přepojování hovorů.

```
[dial-local] ;Kontext
exten => 101,1,Dial(SIP/101) ;Tel. číslo, priorita, akce
exten => 102,1,Dial(SIP/102)
```

Pro provedení změn v souboru je nutné, aby jej Asterisk znovu načetl.

```
~# asterisk -r
reload sip
reload extensions
```

[2, 6]

Na testovacím počítači jsem nainstaloval dva SIP klienty (Yate 5.2.0-1 a Ekiga 4.0.2) a nastavil je, aby se korektně registrovaly v Asterisk.

V souborech */etc/asterisk/manager.conf* a */etc/asterisk/http.conf* jsem nastavil povolení HTTP serveru a přístup přes rozhraní AMI. Pomocí příkazů lze vzdáleně nastavovat parametry ústředny, zjišťovat její stav a ověřovat aktuální dostupnost SIP klientů.

Komunikovat s AMI rozhraním lze pomocí protokolu HTTP metodou POST nebo zabezpečeně pomocí TLS s použitím OpenSSL. V souboru */etc/asterisk/manager.conf* je nutné definovat účet, obsahující přihlašovací jméno, heslo a případná bezpečnostní omezení.

```
[peca]
secret=rfvb
deny=0.0.0.0/0.0.0.0
permit=158.196.0.0/255.255.0.0
permit=127.0.0.1/255.255.255.0
```

V souboru lze také nastavit časové limity spojení. Na konci souboru je nutné zvolit třídy oprávnění ke čtení či zápisu skrze AMI rozhraní.

V souboru */etc/asterisk/http.conf* se povoluje přístup přes HTTP, adresa a port.

Pozměněné soubory musí asterisk znovu načíst:

```
~# asterisk -r
reload manager
reload http
```

[6]

5.3 Instalace SIPp

Program SIPp použiji k ověřování dostupnosti SIP Proxy propojení mezi dvěma PBX. Na testovacím počítači jsem nainstaloval verzi SIPp pro Windows, využívající knihovnu cygwin viz podkapitola 3.4.2. Pro Windows je zkompilována verze 3.2 a je dostupná na adrese „http://sourceforge.net/projects/sipp/files/sipp/3.2/sipp-win32-3.2-setup.exe/download“.

Pro fungování programu bylo potřeba nainstalovat i Cygwin a upravit v inicializačním souboru „startterm.bat“ řádek s cestou ke složce terminfo. V mém případě to bylo:

```
SET TERMINFO=c:\cygwin\usr\share\terminfo
```

[28]

6 Aplikace

K realizaci zadání diplomové práce je možné zvolit libovolný programovací jazyk. Vhodné je zvolit objektově orientovaný programovací jazyk z důvodu snadné rozšiřitelnosti aplikace. K implementaci jsem zvolil programovací jazyk C# a vývojové prostředí Visual Studio 2012 od společnosti Microsoft. [4, 5]

6.1 Návrh

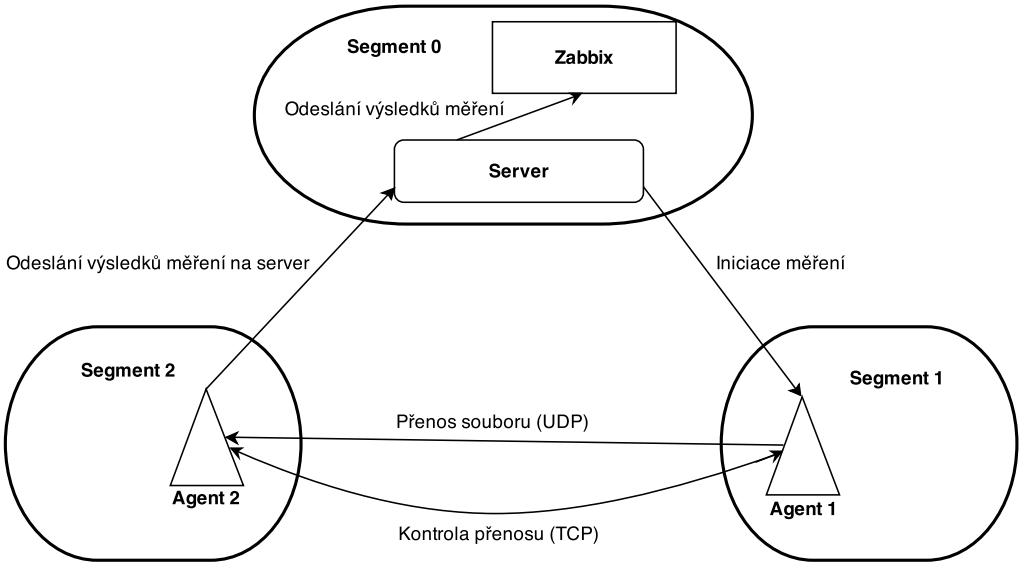
Cílem diplomové práce je aplikace, která bude schopna simulovat hovor IP telefonie mezi dvěma segmenty počítačové sítě. Vyhodnocená kvalita hovoru bude odeslána do dohledového systému Zabbix. Dále pak bude aplikace monitorovat jednotlivé VoIP stanice pomocí metody OPTIONS definované v SIP protokolu a další SIP komponenty.

Základem řešení bude klient/server aplikace. Serverová část bude centrálně řídit celý monitorovací proces. Bude znát IP adresy klientských aplikací rozmístěných v počítačové síti, které jsem dle terminologie systému Zabbix nazval „Agent“. A dle zadaného scénáře bude server iniciovat posílaní dat do jiného segmentu pomocí UDP datagramů, což bude simulovat telefonní hovor. Přenos dat bude trvat dvě minuty, což odpovídá běžnému telefonnímu hovoru. Klient přijímající datagramy následně vyhodnotí kvalitu simulovaného hovoru a výsledek společně s časem měření a identifikací segmentů sítě, mezi kterými došlo k měření, odešle na server. Ten následně odešle naměřená data do Zabbixu ve formátu JSON-RPC. Schéma návrhu této části aplikace je znázorněno na obrázku 3. Kvalita hovoru bude ověřena simulováním čtyř různých kodeků.

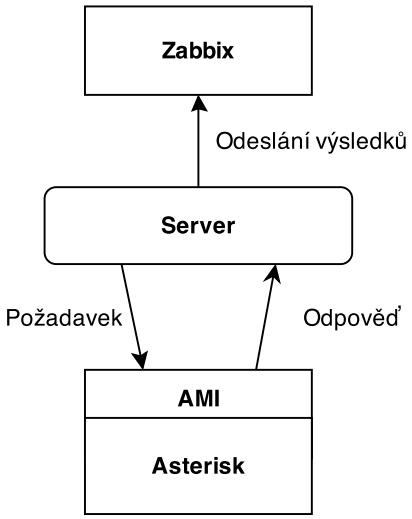
Server bude rovněž ověřovat dostupnost jednotlivých SIP klientů pomocí SIP metody OPTIONS. Ověřování bude probíhat skrze rozhraní AMI pobočkové ústředny Asterisk. Aplikace bude k AMI rozhraní přistupovat pomocí protokolu HTTP a metody POST. Po autentizaci bude poslán požadavek „SIPPeers“ a přijaté informace aplikace projde a vyčte z nich aktuální dostupnost jednotlivých SIP klientů. Data o dostupnosti či nedostupnosti VoIP stanice bude odesílat ve formátu JSON-RPC do systému Zabbix.

Třetí funkcionalitou apliakce bude ověřování dostupnosti SIP Proxy. Dostupnost SIP Proxy a dalších komponentů lze sledovat přímo ze systému Zabbix pomocí programu Ping, případně spuštěním programu SIPp. Aplikace tak bude volat program SIPp s vhodnými parametry, aby navázala SIP komunikaci se SIP Proxy serverem. Pokud server odpoví, je ověřena dostupnost. Pokud žádná odpověď nepřijde, může být Asterisk dynamicky přenastaven, aby použil náhradní SIP Proxy a zkrátil tak dobu výpadku služby.

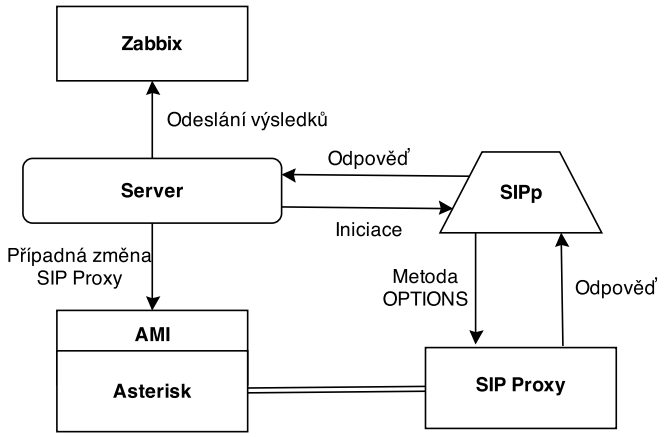
Pro lepší rozšřitilenost a testovatelnost aplikace použiji návrhový vzor injektování závislostí (angl. dependency injection). Tento návrhový vzor vychází z návrhového vzoru Inversion of control. Pro implementaci v jazyce C# použiji kontejner IoC/DI Windsor.castle. Aplikace bude rovněž, tam kde to bude vhodné, vícevláknová. Například každá ze tří funkcí aplikace bude spuštěna v samostatném vláknu.



Obrázek 3: Schéma části navrhované aplikace monitorující kvalitu hovoru



Obrázek 4: Schéma části navrhované aplikace monitorující dostupnost SIP klientů



Obrázek 5: Schéma části navrhované aplikace monitorující dostupnost SIP Proxy

6.1.1 Komunikační protokol

Pro komunikaci mezi serverem a agentem a mezi agenty samotnými je nutné zvolit nějaký komunikační protokol. K tomuto účelu postačí jednoduchý textový formát, kde bude název proměnné: proměnná a jednotlivé údaje budou odděleny středníkem. Znakem ukončení řádku je indikován konec dotazu „\n“.

Server bude posílat agentu A tyto informace: název akce, IP adresa agenta B a název kodeku.

```
Action:TestCall;IPAddressB:192.168.201.56;Codec:G.711\n
```

Agent odešle na přidělenou IP adresu protistraně název akce, IP adresu serveru, název kodeku a celkový počet UDP datagramů, které má očekávat.

```
Action:TestCall;ServerIPAddress:192.168.1.2;Codec:G.711;
   Count:6000\n
```

Poté začne posílat agent A agentu B UDP datagramy po dobu dvou minut rychlostí 96 kbit/s, což odpovídá datovému toku při hovoru kódovaném kodekem G.711. Agent B zkontroluje ztráty při přenosu dat a dobu přenosu a na server odešle výsledek měření. Název akce, IP adresu agenta A, od kterého simulovaný hovor přijal, název kodeku, výslednou hodnotu R vypočtenou z E-modelu, výsledek kvality hovoru dle stupnice MOS, začátek a konec měření.

```
Action:TestCall;IPAddressA:192.168.78.6;Codec:G.711;R:71,3;
   MOS:3,66;Start:20140203122315;End:20140203122327\n
```

Získané informace o kvalitě hovoru mezi agenty A a B odešle server ve formátu JSON-RPC dohledovému systému Zabbix. Odeslané informace budou obsahovat IP adresy obou agentů, název testovacího kodeku, hodnotu parametru R, kvalitu hovoru dle stupnice MOS, začátek měření a konec měření.

Ke komunikaci s dohledovým systémem Zabbix bude použit komunikační protokol JSON-RPC viz podkapitola 2.2.9.

Komunikace s AMI rozhraním Asterisk probíhá pomocí protokolu HTTP a metody POST, případně zabezpečeně protokolem TLS prostřednictvím OpenSSL.

6.2 Monitorování kvality hovoru

Pro monitorování kvality hovoru je nutné spustit aplikaci na dvou různých stanicích. Předpokládá se, že budou spuštěny v oddělených segmentech, mezi kterými je potřeba kontrolovat kvalitu hovoru. Není však samozřejmě překážkou kontrolovat kvalitu hovoru v rámci stejného segmentu počítačové sítě.

Aplikace je pro tento účel schématicky rozvržena do tří komponent zobrazných na obrázku 3.

První částí je server, který zajišťuje:

1. iniciaci měření odesláním příkazu,
2. příjem výsledků měření,
3. odeslání naměřených hodnot do Zabbix serveru,
4. vytvoření nového prvku v Zabbix serveru, pokud zatím nebyl vytvořen.

Seznam IP adres agentů je uložen v souboru „Agents.txt“, kdy každá IP adresa je na samostatném řádku oddělená znakem „\n“.

Aplikace pak náhodně vybírá vždy dva agenty, mezi kterými jsou iniciována měření odesíláním příkazů přílušnému agentu, které obsahují název akce, IP adresu agenta, který bude přijímat testovací data, a zvolený kodek, podle jehož parametrů jsou strukturována a odesílána data simulující telefonní hovor. Název akce je specifikován z důvodu případného pozdějšího rozšíření aplikace o další funkce. V tomto případě je zvolen název akce „TestCall“. Definice příkazů a specifikace komunikačního protokolu je popsána v podkapitole 6.1.1.

Agent, který od serveru příkaz příjme, nejprve dle zadaného kodeku spočítá celkový počet datagramů, které odešle v rámci testovacího hovoru. Poté se pokusí připojit na TCP soket druhého agenta, jehož IP adresu získal z příkazu a odešle agentovi stanovený příkaz, který obsahuje IP adresu serveru, použitý kodek a celkový počet datagramů.

Pokud je druhý agent dostupný, začne první agent odesílat datagramy pomocí UDP. Velikost datagramů a jejich počet za vteřinu je dán typem kodeku, jenž je simulován, a dále je ovlivněn přidáním RTP hlavičky. Výpočet parametrů včetně RTP hlavičky je uveden a popsán v podkapitole 4.3.

Po dokončení odesílání datagramů je odeslán poslední o velikosti jeden bajt, který indikuje ukončení odesílání. Agent, který datagramy přijal porovná počet přijatých s počtem očekávaných datagramů a vypočítá ztrátovost přenosu. Poté vypočítá kvalitu hovoru pomocí E-modelu, viz podkapitola 4.1, na základě ztrátovosti datagramů a použitého kodeku. Vypočítanou hodnotu R-faktoru, společně s IP adresou agenta odesílajícího data, použitým kodekem a časem měření, odešle na server.

Server získaná data z měření kvality hovoru zpracuje a odešle je do Zabbix serveru pomocí programu Zabbix sender. Příkaz spouštějící Zabbix sender může například vypadat následovně:

```
zabbix_sender.exe -z 158.196.244.165 -p 10051 -s
   192.168.1.2-10.0.1.2 -k G.711R -o 88.1 -I
   158.196.195.181
```

Parametry jsou popsány v podkapitole2.2.7. V tomto případě se jedná o aktualizaci údajů o kvalitě hovoru mezi agenty s IP adresami 192.168.1.2 a 10.0.1.2. Monitorovaný prvek je pojmenován spojením IP adres agentů. Klíč monitorované položky je název kodeku a písmeno „R“, značící R-faktor. Hodnota klíče je pak 88.1. Posledním parametrem je specifikace IP adresy, ze které jsou údaje odesílány. Tato IP adresa musí být v Zabbix serveru uvedena u aktualizované položky mezi „Allowed hosts“. Problém, způsobený přidělováním odlišné IP adresy při každém připojení k univerzitní počítačové síti, je popsán v kapitole 6.5.1.

Následně je celý postup opakován pro další kodeky, které jsou celkem čtyři a jsou testovány v tomto pořadí:

1. PCM - ITU-T G.711
2. MP-MLQ - ITU-T G.723.1
3. ACELP - GSM 06.60
4. CS-ACELP - ITU-T G.729-A + VAD

Parametry kodeků jsou popsány v podkapitole 4.3.

Po otestování všemi kodeky aplikace náhodně vybere další dva agenty, u kterých proběhne stejné testování.

6.2.1 Simulační režim

Pro účely testování aplikace je implementován simulační režim, ve kterém budou simulovány ztráty při přenosu datagramů. Implementace tohoto režimu byla nutná, protože v testovací topologii bylo obtížné zajistit ztrátovost při přenosu dat.

Ztrátovost je zajištěna tím, že po vypočítání celkového počtu datagramů, které budou odeslány, je tento počet vynásoben číslem z intervalu $\langle 0; 1 \rangle$ vyjadřujícím definovanou ztrátovost.

```
counter = PacketCount − (int)((double)PacketCount ∗ lostsPercentIndex);
```

Výpis 1: Simulace ztrátovosti

Agentu, který bude data přijímat, je pak sdělen vyšší počet datagramů, které má očekávat, než který je mu skutečně odeslán. Tento rozdíl simuluje ztrátovost při přenosu.

6.3 Monitorování SIP klientů

Pro monitorování SIP klientů využívá aplikace pobočkovou ústřednu Asterisk, ke které přistupuje prostřednictvím rozhraní AMI. Pomocí HTTP POST požadavků se nejprve autentizuje s příhlašovacím jménem a heslem, které je nastaveno v souboru */etc/asterisk/manager.conf* - viz podkapitola5.2.2. Ke kódování jsem použil rawman, aby odpověď serveru přišla jako HTTP odpověď.

Požadavek k autentizaci vypadá následovně:

```
http://158.196.244.165:8088/rawman?action=login&username=
   peca&secret=rfvb
```

Po úspěšné autentizaci je odpověď serveru:

```
Response: Success
Message: Authentication accepted
```

Aplikace z odpovědi zkontroluje, zda bylo přihlášení úspěšné a může odeslat další požadavek, kterým je žádost o vypsání seznamu SIP klientů a jejich stavu požadavkem „SIPpeers“.

```
http://158.196.244.165:8088/rawman?action=SIPpeers
```

V odpovědi je výpis všech SIP klientů, kteří jsou definovaní v souboru „sip.conf“. Příklad výpisu registrovaného SIP klienta:

```
Event: PeerEntry
Channeltype: SIP
ObjectName: 101
ChanObjectType: peer
IPaddress: 158.196.195.204
IPport: 5060
Dynamic: yes
AutoForcerport: yes
Forcerport: no
AutoComedia: no
Comedia: no
VideoSupport: no
TextSupport: no
ACL: no
Status: OK (49 ms)
RealtimeDevice: no
Description:
```

Aplikace tak ke každému klientovi získá jeho název, IP adresu a port, status a zpoždění v milisekundách. Tyto informace aplikace použije k vytvoření parametrů pro Zabbix sender. Název je shodný s názvem prvku v dohledovém systému Zabbix. Klíče jsou IP adresa:Port, status a zpoždění. Hodnotami jsou pak příslušné údaje (158.196.195.204:5060,

OK, 49). Aplikace tak musí spustit Zabbix sender celkem třikrát, vždy pro každý klíč samostatně.

Pokaždé je, stejně jako v ostatních částech aplikace, kontrolováno, zda bylo odeslání údajů úspěšné ověřením sekvence „Failed: 0“ v odpovědi. Pokud odeslání nebylo úspěšné, pokusí se aplikace vytvořit prvek s příslušným názvem v Zabbix serveru pomocí Zabbix API. Viz podkapitola 6.5.

Jednotliví SIP klienti jsou ve výpisu odděleni prázdným řádkem a konec výpisu aplikace detekuje přítomností řádku „Event: PeerlistComplete“.

Vyhodnocení údajů a případnou notifikaci uživatelů provádí Zabbix server prostřednictvím příslušných triggerů. Například dostupnost SIP klienta je ověřována ověřováním hodnoty klíče „Status“. Trigger porovnává jestli je hodnota klíče „OK“. Je-li text odlišný, například UNKNOWN, je klient vyhodnocen jako nedostupný.

Další možností, jak monitorovat dostupnost SIP klienta je kontrolou IP adresy. Pokud je ve výpisu z Asterisk AMI „IPaddress: -none-“, indikuje to aplikaci nedostupnost SIP klienta. Trigger v Zabbix serveru tak také kontroluje, zda není hodnota klíče „IP adresa“ rovna „-none-“.

6.4 Monitorování SIP Proxy

Aplikace spouští program SIPp s parametry definujícími, že SIPp bude v roli klienta posílat na IP adresu „158.196.244.165“ jednu zprávu OPTIONS[2]. Popis parametrů je uveden v podkapitole 3.4.2 a instalace SIPp v podkapitole 5.3.

```
sipp.exe -sn uac 158.196.244.165 -sf OPTIONS_recv_200.xml -
   i 158.196.194.154 -m 1
```

Po vyhodnocení statistiky se ve výpisu programu objeví, mimo jiné, i název serveru.

```
Server: Asterisk PBX 11.7.0
```

Pokud bude SIP služba nedostupná, neobjeví se ani položka „Server:“. V takovém případě aplikace odešle do systému Zabbix pomocí programu Zabbix sender aktualizaci prvku SIP Proxy a Zabbix o události notifikuje uživatele pomocí e-mailu a SMS zprávy.

IP adresu SIP Proxy lze nastavit v konfiguračním souboru aplikace.

6.5 Komunikace se Zabbix

Serverová část mé aplikace musí nejprve vytvořit příslušný prvek v dohledovém systému Zabbix, ke kterému může následně odesílat data o stavu a další parametry. Tyto prvky aplikace vytváří dynamicky vždy v případě, kdy se nepodaří odeslat údaje pomocí programu Zabbix sender. Aplikace z výpisu programu Zabbix sender ověří, zda obsahuje textový řetězec „Failed: 0“. Pokud odesílání selhalo, dojde k vytvoření příslušného prvku. K vytvoření prvku používá aplikace příkazy ve formátu JSON-RPC a odesílá je pomocí

[2]XML soubor, který jsem použil, je dostupný online na stránce http://tomeko.net/other/sipp/scenarios/OPTIONS_recv_200.xml

HTTP. K serializaci a deserializaci objektu je použit framework JSON.NET od Newtonsoft[3].

Jako příklad vytvoření JSON-RPC příkazu uvádím příkaz, který nese metodu „user.login“, jež se používá k autentizaci pomocí přihlašovacího jména a hesla.

```
public string Login(string user, string password)
        {
            var loginRequest = new LoginRequestDTO
            {
                Id = "1",
                Jsonrpc = "2.0",
                Method = "user.login",
                Params = new LoginRequestParamsDTO
                {
                    Password = password,
                    User = user
                }
            };
            var requestString = Serializer.Serialize(loginRequest);
            var responseString = SendRequest(requestString);
            return Serializer.Deserialize<LoginResponseDTO>(responseString).Result;
        }
```

Výpis 2: Příklad JSON-RPC metody

Metoda vrátí náhodně vygenerovanu sekvenci znaků v poli „Auth“, která je pak aplikací použitá k autentizaci dalších metod až do doby odhlášení se.

6.5.1 Problém s „Allowed hosts“

K odesílání dat do Zabbix serveru využívá aplikace program Zabbix sender. Aby bylo možné hodnotu klíče v položce některého prvku v Zabbix serveru aktualizovat, musí to být položka typu „Zabbix trapper“. Navíc je nutné, aby byla IP adresa počítače, na kterém je aplikace spuštěna, uvedena u položky jako „Allowed hosts“. Tím však vyvstává nutnost, spouštět aplikaci na počítači se statickou IP adresou. Zabbix bohužel v současné verzi neumožňuje z bezpečnostních důvodů uvést jako „Allowed hosts“ například IP adresu celé podsítě. Ve verzi Zabbix 2.2 lze však použít k definici povolených hostitelů uživatelská makra, kterými lze nastavit rozsah IP adres. [15]

V současné verzi však tento nedostatek vyřešen není. Při testování aplikace tak nastává problém s IP adresou, která je univerzitním DHCP serverem přidělena testovacímu počítači při každém připojení vždy jiná. Je tak nutné před testováním změnit IP adresu „Allowed hosts“ u každé položky v šablonách Zabbix serveru.

[3] http://james.newtonking.com/json

6.6 Testování

Pro otestování aplikace jsem použil svou testovací topologii popsanou v kapitole 5. Topologie se skládá z virtuálního serveru na kterém je spuštěn Asterisk a Zabbix, a dvou počítačů.

Na každém počítači je spuštěna vyvíjená aplikace. Na jednom počítači je spuštěn server a agent A a na druhém je agent B. Na každém z počítačů je pak spuštěn jeden SIP klient, který je korektně registrován v PBX Asterisk. V konfiguračních souborech jsou nastaveny správné IP adresy agentů, serveru, Zabbix i Asterisk.

V dohledovém systému Zabbix jsem před začátkem testování musel přidat IP adresu serveru aplikace k „Allowed hosts“ z důvodu popsaného v kapitole 6.5.1.

Aplikace po spuštění začala testovat kvalitu hovoru mezi oběma počítači.

V dalším vlákně server aplikace kontroloval dostupnost SIP klientů, které jsem pro testovací účely vypínal a zapínal a zjišťoval tak funkčnost kontroly jejich dostupnosti. Nejrychlejším indikátorem dostupnosti byla kontrola IP adresy SIP klienta ve výpisu z Asterisk.

V posledním vlákně server aplikace zjišťuje dostupnost SIP Proxy, kterou jsem pro testovací účely zvolil již běžící pobočkovou ústřednu Asterisk.

Při testování vše fungovalo a spolupracovalo.

7 Závěr

Porovnáním dohledových systémů vhodných pro monitorování IP telefonie a SIP komponent jsem dospěl k závěru, že jsou oba pro tyto účely vhodné, avšak zvolil jsem systém Zabbix z důvodu kompatibility výstupů práce s běžícím projektem na katedře telekomunikační techniky. K notifikaci o událostech posílá systém Zabbix e-maily a SMS zprávy. Pro odesílání e-mailů jsem využil školni SMTP server „smtp.vsb.cz“. SMS zprávy jsou odesílány pomocí GSM modulu Huawei E303, který je připojen do USB rozhraní.

Dohledový systém Zabbix je připraven na dynamické vytváření monitorovaných prvků a jejich přiřazování ke skupinám prvků a šablonám. Rovněž jsou připraveny způsoby notifikace administrátorského účtu.

Pro testovací účely provozuji také částečně nastavenou pobočkovou ústřednu Asterisk a dva SIP klienty.

Ke sběru dat o SIP klientech jsem implementoval ve své aplikaci funkcionalitu, která získá údaje o SIP účtech a jejich aktuální dostupnost. Tento způsob je vhodnější, než posílat zprávy OPTIONS na SIP klienty, protože tím je ověřena jen dostupnost samotného klienta a nikoli jeho korektní zaregistrování v PBX Asterisk. Získaná data jsou pak podle názvu SIP účtu roztříděna a ke každému SIP účtu je v dohledovém systému Zabbix vytvořen prvek. Poté jsou k prvkům a jeho položkám odeslány údaje o dostupnosti a době zpoždění mezi PBX a SIP klientem, pomocí programu Zabbix sender. Data jsou pak v systému Zabbix vyhodnocena a je vyvolána příslušná akce, která zprostředkuje notifikace uživatele Zabbix serveru.

K monitorování kvality hovoru, mezi SIP uživateli je jako součást aplikace implementován tzv. agent, který na základě požadavku serveru simuluje telefonní hovor s jiným agentem. Hovory jsou simulovány pro čtyři kodeky (G.711, G723.1, G.729-A a GSM). Každý kodek využívá při přenosu jinou šířku pásma a má jiné parametry ovlivňující výslednou kvalitu hovoru. Pro tyto čtyři kodeky jsou pak vypočítány hodnoty R-faktoru podle E-modelu. Číselné vyjádření kvality hovoru je pro každý spoj a použitý kodek odesláno do Zabbix serveru rovněž pomocí programu Zabbix sender. Ještě předtím jsou však vytvořeny prvky v Zabbix serveru reprezentující spoj mezi dvěma segmenty počítačové sítě.

Poslední funkcionalitou aplikace je ověřování dostupnosti služeb SIP Proxy pomocí programu SIPp, který odesílá SIP zprávy OPTIONS na příslušné SIP Proxy a ověřuje tak SIP trunk mezi dvěma ústřednami. Údaje o dostupnosti jsou odesílány do systému Zabbix a v případě nedostupnosti SIP Proxy následuje notifikace SMS zprávou.

Výsledná aplikace je implementována tak, aby byla snadno testovatelná a snadno rozšiřitelná. To umožňuje, aby výsledek mé závěrečné práce byl použit v praxi a byl případně rozšířen o další funkcionalitu.

Jednou z možností rozšíření aplikace je změna konfigurace pobočkové ústředny Asterisk, úpravou souboru „extensions.conf“, v případě zjištění nedostupnosti SIP Proxy. Konfigurace by mohla být změněna tak, aby PBX začala používat záložní SIP Proxy a tím by se zkrátila doba výpadku.

Bc. Petr Vůjtek

8 Reference

[1] VOZŇÁK, Miroslav. Voice over IP. dotisk 1. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2008, 176 s. ISBN 978-80-248-1828-3.

[2] L. Madsen, J. Meggelen and R. Bryant, Asterisk: The Definitive Guide. O Reilly, 734 p. , 2011, ISBN 978-14-493-0680-9.

[3] V. Faltinsen, G. Vindheim, Framework conditions and requirements for network monitoring in campus networks. Technical Report GN3-NA3-T4-UFS128, 29 p., TERENA, Amsterdam, 2011. Dostupné z: http://www.terena.org/activities/campus-bp/pdf/gn3-na3-t4-ufs128.pdf

[4] TROELSEN, Andrew W. Pro C# 5.0 and the .NET 4.5 Framework. New York: Apress, 2012. 6th ed. ISBN 978-1-4302-4234-5.

[5] BETTS, Dominic, Grigori MELNIK, SIMONAZZI a Mani SUBRAMANIAN. MICROSOFT. Dependency Injection with Unity: Patterns & Practices [online]. Redmond, WA, USA, 2013 [cit. 2014-03-02]. ISBN 78-1-62114-028-3. Dostupné z: http://www.microsoft.com/en-us/download/confirmation.aspx?id=39944

[6] Asterisk Project Wiki. Asterisk Project [online]. [cit. 2014-04-11]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Home

[7] BARTH, Wolfgang. Nagios: system and network monitoring. U.S. ed. San Francisco: No Starch Press, c2006, 462 p. ISBN 978-159-3270-704.

[8] NAGIOS. Nagios Library [online]. 2009-2013 [cit. 2013-02-02]. Dostupné z: http://library.nagios.com/

[9] KOCJAN, Wojciech. Learning Nagios 3.0: a detailed tutorial to setting up, configuring, and managing this easy and effective system monitoring software. Birmingham, U.K.: Packt Pub., c2008, v, 301 p. From technologies to solutions. ISBN 978-1-84719-518-0.

[10] Nagios Exchange: check_calls. In: Nagios [online]. 2012, 25.4.2012 [cit. 2013-02-03]. Dostupné z: http://exchange.nagios.org/directory/Plugins/Network-Protocols/*-VoIP/SIP/check_calls/details

[11] INTEL CORPORATION, Hewlett-Packard Company, NEC Corporation, Dell Computer Corporation. IPMI: Intelligent Platform Management Interface Specification v1.5. Delaware (USA), 2004. Dostupné z: http://download.intel.com/design/servers/ipmi/IPMIv1_5rev1_1_6-1-04_markup.pdf

[12] OLUPS, Rihards. Zabbix 1.8 network monitoring: monitor your network's hardware, servers, and Web performance effectively and efficiently. Birmingham, U.K.: Packt Pub., 2010, vii, 410 p. ISBN 978-1-847197-68-9.

[13] Installation Zabbix: Zabbix documentation. ZABBIX SIA. Zabbix [online]. 2012, 2012/10/09 12:16 [cit. 2013-02-16]. Dostupné z: https://www.zabbix.com/documentation/2.0/manual/installation/install

[14] Zabbix documentation: Zabbix documentation 2.0. ZABBIX SIA. Zabbix [online]. 2012, 2012/10/09 12:16 [cit. 2013-03-07]. Dostupné z: https://www.zabbix.com/documentation/2.0

[15] Zabbix documentation: Zabbix documentation 2.2. ZABBIX SIA. Zabbix [online]. 2013, [cit. 2014-04-17]. Dostupné z: https://www.zabbix.com/documentation/2.2

[16] CROCKFORD, Douglas. RFC 4627. The application/json Media Type for JavaScript Object Notation (JSON) [online]. USA: The Internet Society, July 2006 [cit. 2013-01-23]. Dostupné z: http://www.ietf.org/rfc/rfc4627.txt

[17] Handley, et al. RFC 2543. SIP: Session Initiation Protocol [online]. USA: The Internet Society, March 1999 [cit. 2013-01-27]. Dostupné z: http://www.ietf.org/rfc/rfc2543.txt

[18] DONOVAN, Steve. RFC 2976. The SIP INFO Method [online]. USA: The Internet Society, October 2000 [cit. 2014-02-02]. Dostupné z: http://www.ietf.org/rfc/rfc2976.txt

[19] Rosenberg, et. al. RFC 3261. SIP: Session Initiation Protocol [online]. USA: The Internet Society, July 2002 [cit. 2013-01-27]. Dostupné z: http://www.ietf.org/rfc/rfc3261.txt

[20] J. Rosenberg, H. Schulzrinne. RFC 3262. Reliability of Provisional Responses in the Session Initiation Protocol (SIP) [online]. USA: The Internet Society, June 2002 [cit. 2014-02-02]. Dostupné z: http://www.ietf.org/rfc/rfc3262.txt

[21] ROACH, Adam. RFC 3265. Session Initiation Protocol (SIP)-Specific Event Notification [online]. USA: The Internet Society, June 2002 [cit. 2014-02-02]. Dostupné z: http://www.ietf.org/rfc/rfc3265.txt

[22] ROSENBERG, Jonathan. RFC 3311. The Session Initiation Protocol (SIP) UPDATE Method [online]. USA: The Internet Society, September 2002 [cit. 2014-02-03]. Dostupné z: http://www.ietf.org/rfc/rfc3311.txt

[23] Campbell, Rosenberg, Schulzrinne, Huitema, Gurle. RFC 3428. Session Initiation Protocol (SIP) Extension for Instant Messaging [online]. USA: The Internet Society, December 2002 [cit. 2014-02-03]. Dostupné z: http://www.ietf.org/rfc/rfc3428.txt

[24] SPARKS, Robert. RFC 3515. The Session Initiation Protocol (SIP) Refer Method [online]. USA: The Internet Society, April 2003 [cit. 2014-02-03]. Dostupné z: http://www.ietf.org/rfc/rfc3515.txt

[25] P. Faltstrom, M. Mealling. RFC 3761. ENUM [online]. USA: The Internet Society, April 2004 [cit. 2013-02-01]. Dostupné z: http://www.ietf.org/rfc/rfc3761.txt

[26] NIEMI, Aki. RFC 3903. Session Initiation Protocol (SIP) Extension for Event State Publication [online]. USA: The Internet Society, October 2004 [cit. 2014-02-03]. Dostupné z: http://www.ietf.org/rfc/rfc3903.txt

[27] SIPSak - Linux man page [online]. [cit. 2014-01-30]. Dostupné z: http://linux.die.net/man/1/sipsak

[28] JACQUES, Olivier a Robert DAY. SIPp: Documentation SIPp v3.2 [online]. 2013, 10.11.2013 [cit. 2014-01-30]. Dostupné z: http://sipp.sourceforge.net/doc3.2/reference.html

[29] CHIA, Billy. How to Install Asterisk 11 on Ubuntu 12.04 LTS. In: Digium, Inc. [online]. 2012, November 14 [cit. 2014-02-05]. Dostupné z: http://blogs.digium.com/2012/11/14/how-to-install-asterisk-11-on-ubuntu-12-4-lts

[30] Recommendation ITU-T G.107. The E-model: a computational model for use in transmission planning. 12/2011 [cit. 2014-02-12]. Dostupné z: http://www.itu.int/rec/T-REC-G.107.1

[31] M. Voznak, Recent Advances in Speech Quality Assessment, The International Conference on Advanced Engineering - Theory and Applications 2013, (Keynote Speech) In SPRINGER Lecture Notes in Electrical Engineering, Volume 282, 2014, pp. 1-14, Ho Chi Minh City, Vietnam, December 11-13, 2013

[32] M. Voznak, F. Rezac, J. Rozhon, Speech Quality Assessment Using Computational E-Model and its Implementation, INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTERS IN SIMULATION, Issue 3, Volume 7, 2013, ISSN 1998-0159, pp. 277-285.

[33] M. Voznak, A Kovac, M. Halas, Effective Packet Loss Estimation on VoIP Jitter Buffer, Computing in Networks 2012, In SPRINGER Lecture Notes in Computer Science, LNCS 7291/2012, Prague, May 2012, pp. 157-162, ISBN 978-3-642-30038-7, ISSN 0302-9743

[34] M. Voznak, J. Rozhon, F. Rezac, J. Slachta, Real-Time Speech Quality Monitoring Using Non-Intrusive Method, RECENT RESEARCHES in CIRCUITS, COMMUNICATIONS and SIGNAL PROCESSING [online]. Milan, January 9-11, 2013, pp. 43-48. Dostupné z http://www.wseas.us/e-library/conferences/2013/Milan/CSVL/CSVL-07.pdf

[35] K. Tomala, L. Macura, M. Voznak, J. Vychodil, Monitoring the Quality of Speech in the Communication System BESIP, In Proc. 35th International Conference on Telecommunication and Signal Processing, Prague, July 3-4, 2012, pp. 255-258, ISBN 978-14673-1116-8 (WoS, SCOPUS , IEEE-Xplore).

[36] Recommendation ITU-T G.711. Wideband embedded extension for ITU-T G.711 pulse code modulation. 09/2012 [cit. 2014-03-25]. Dostupné z: http://www.itu.int/rec/T-REC-G.711.1

[37] Recommendation ITU-T G.113. Transmission impairments due to speech processing. 11/2007 [cit. 2014-03-25]. Dostupné z: http://www.itu.int/rec/T-REC-G.113

[38] Recommendation ITU-T P.800. Methods for objective and subjective assessment of quality. 08/1996 [cit. 2014-02-26]. Dostupné z: http://www.itu.int/rec/T-REC-P.800

[39] ETSI: Technical Specification TS 101 318. Using GSM speech codecs within ITU-T Recommendation H.323. 1998-08 [cit. 2014-04-04]. Dostupné z: http://www.etsi.org/deliver/etsi_ts/101300_101399/101318/01.01.01_60/ts_101318v010101p.pdf

Poděkování

Tato práce byla vypracována s podporou projektu Rozvoj lidských zdrojů ve výzkumu a vývoji moderních soft computingových metod a jejich praktického využití, reg. č. CZ.1.07/2.3.00/20.0072 podpořeného Operačním programem Vzdělávání pro konkurenceschopnost, financovaného ze strukturálních fondů EU a státního rozpočtu ČR.